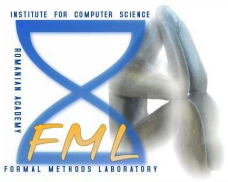

Formal Methods Laboratory

Romanian Academy

FML Technical Report

TR no. FML-12-02

Printed:
December 2012

Abstract Continuation Semantics for Asynchronous Concurrency

G.Ciobanu and E.N.Todoran

Prepared by
Formal Methods Laboratory.

Series of
Technical Reports (Institute of Computer Science Iaşi)

ISSN 1842 - 1490

FML is the Formal Methods Laboratory from the Institute of Computer Science of the Romanian Academy and is located in Iaşi. It is a research institution aimed at developing new formalisms for challenging open problems in computer science, systems biology and other emerging research fields. FML is headed by Dr. Gabriel Ciobanu.

IIT is the Institute of Computer Science of the Romanian Academy and is concerned with basic research in computer science.

Romanian Academy of Science is the supreme forum of science from Romania.
Calea Victoriei 125, Sector 1, 010071, Bucharest, ROMANIA.
Telephone: +40 21 2128640
Telefax: +40 21 2116608
Web: `http://www.academiaromana.ro/`

Printed in Romania and reproduced directly from the authors original copy.

Blvd. Carol I, nr.8, 700505, Iaşi, ROMANIA (RO)
Institute of Computer Science, Romanian Academy, branch Iaşi.
Telephone/Telefax: +40 232 241708
Web: `http://iit.iit.tuiasi.ro/fml`
Contact: `gabriel@iit.tuiasi.ro`

Series of Technical Reports (Institute of Computer Science Iaşi)

ISSN 1842 - 1490

Abstract Continuation Semantics for Asynchronous Concurrency

by

Gabriel Ciobanu [1,2]
Institute of Computer Science,
Romanian Academy, Iaşi
gabriel@iit.tuiasi.ro

Eneia Nicolae Todoran
Technical University of Cluj-Napoca
Department of Computer Science
400027 Cluj-Napoca, Romania
Eneia.Todoran@cs.utcluj.ro

ABSTRACT

The technical report presents a denotational semantics of a simple asynchronous language designed with metric spaces and contin- uation semantics for concurrency. The denotational model satisfies an optimality criterion that we call *weak abstractness*.

[1]Head of Formal Methods Laboratory (FML)
[2]Contact person

Contents

1 Introduction

The technique of continuations is a classical tool in denotational semantics. In [27] we introduced a variant of this technique that we named "continuation semantics for concurrency" (CSC). In the CSC approach a continuation is an application-specific structure of computations (partially evaluated denotations) rather than a function to some answer type as in the classic technique of continuations [26]. The structure of continuations in CSC can be designed by combining the concept of a *stack* and the concept of a *multiset*. Sequential composition can be modeled by structuring continuations as stacks of computations. Also, parallel composition can be modeled by structuring continuations as multisets of computations. In order to model a general combination of sequential and parallel composition continuations can be structured as trees of computations [27]; such a tree implements the concept of a *cactus stack* [7], i.e., a stack with multiple tops that can be active concurrently. Unlike other models of concurrency [22], [3], [9], in the CSC approach the final yield of the denotational mapping is a simple collection of observations and all control and communication concepts are modeled as operations manipulating continuations.

In this paper we present a method which can be used to reason about the behavior of concurrent programs in denotational models designed with CSC and metric spaces. The metric approach to semantics is presented in the monograph [3]. For illustration purposes we consider a simple asynchronous language, embodying the paradigm of asynchronous communication introduced in [8]. For the language under consideration we prove some concurrency laws, such as the associativity and comutativity of parallel composition. The proof method relies on the identification of behavioral invariants expressed as relations between continuation structures and the use of contraction in complete metric spaces.

To the best of our knowledge there is no published full abstractness result for a denotational model of a concurrent language designed with continuations. Also, we do not know whether a fully abstract domain of CSC exists. In this paper we investigate a weaker optimality criterion - that we call *weak abstractness* - which can be estblished for CSC.

In CSC a continuation is a language-specific structure of computations. In general, a continuation may contain computations that are not denotable by any program statement. The research reported in this paper focuses on the class of continuations that contain only denotations of the program statements, which we call the class of *denotable continuations*. Evaluation with respect to denotable continuations represents an invariant of the denotational semantics and ensures its consistency just because the initial continuation is empty and the denotational semantics adds to the continuation only denotations of the language constructs. Some properties in CSC can only be proved for the class of denotable continuations (this was already investigated in our previous work [12, 11]). In this paper we also use the concept of a denotable continuation to define an (weak) abstractness criterion that can be established for CSC.

The weak abstractness criterion we define and establish within the mathematical framework of complete metric spaces. We follow the approach advocated in [3]. The domain for CSC is defined based on a certain domain equation which can be solved by using the general

method introduced in [1].

1.0.1 A simple asynchronous formalism

We investigate a simple language $\mathcal{L}$ embodying the paradigm of asynchronous communication introduced in [8]. Instances of this paradigm include concurrent constraint programming [25], and also in other languages like dataflow, asynchronous CCS and asynchronous CSP. The importance of asynchronous communication and the relation between synchronous and asynchronous communication is investigated in [15] [21] [31]. A simulation of synchronous interaction by means of asynchronous interaction is not always possible. However, asynchronous communication is easier to implement and it represents the basic interaction mechanism in many distributed systems, including most Internet and Web applications. The research reported in this paper starts from a denotational semantics designed with metric spaces and CSC for $\mathcal{L}$.

A denotational semantics designed with metric spaces and CSC for $\mathcal{L}$ was first presented in [27]. There is only one minor difference between the denotational model given in this paper and the one given in [27]. In order to obtain a weak abstractness result in this paper we use the notion of a *consistent* continuation, and we define the behavior such that any $\mathcal{L}$ program blocks immediately if it is evaluated with respect to a nonconsistent continuation. In the CSC approach the space of computations may be divided conceptually into an active computation and the rest of computations which are grouped into the continuation. In the denotational model designed with CSC for $\mathcal{L}$ a continuation is a tree of computations. Intuitively, a continuation is consistent if the active computation is a leaf in the tree of computations representing the continuation. The consistency condition is simple and it represents an invariant property that is preserved by the denotational semantics, just because the initial continuation is consistent and each equation defining the denotational semantics preserves the consistency condition. No semantic property is affected by this design decision. Moreover, the proofs are similar for consistent continuations and trivial for nonconsistent continuations. In addition, the denotational model given in this paper is both *correct* (the correctness condition was already demonstrated in [27]) and *weakly abstract* - a notion that is introduced in this paper.

1.0.2 Concurrency laws in continuation semantics

In Section 5 we provide some results from [12, 11]. The results are trivial for nonconsistent continuations. Also, the proofs for consistent continuations are similar to the proofs given in [12, 11], hence they are ommited here. The results given in Section 5 represent concurrency laws that are satisfied by the semantic operators of $\mathcal{L}$, such as the associativity and commutativity of parallel composition. In general, concurrency laws can only be establised for the class of denotable continuations. We also investigate the semantic properties in the metric completion of the class of denotable continuations, which we call the *domain of denotable continuations*. We establish the result that all concurrency laws also hold in the domain of denotable continuations.

Continuation-based models rely on manipulations of higher-order functions. It may be difficult to reason directly in terms of higher-order functions. Therefore $\mathcal{L}$ includes a *left merge* operator that allows us to establish a finite axiomatization of the parallel composition (or *merge*) operator. Any non-recursive asynchronous concurrent program is thus provably equivalent to a corresponding nondeterministic sequential program. The approach is inspired by classic process algebra theories [20, 6]; this approach is adapted by us to a continuation-based framework (see also [12, 11]).

Each semantic property, also called a *law* here, can be proved by identifying a corresponding invariant of the computation, as a relation between CSC continuation structures. The identification of semantic properties from the invariants of the computation is common in classic bisimulation semantics [20]. In [12, 11] we have shown that this idea can be adapted to a continuation-based denotational framework. The basic idea is that one can use arguments of the kind '$\varepsilon \leq \frac{1}{2} \cdot \ \cdot \varepsilon \Rightarrow \varepsilon = 0$' (which are standard in metric semantics [3]), where ε is the distance between two behavioraly equivalent continuations, before and after a computation step, respectively. The effect of each computation step is given by the $\frac{1}{2}$ contracting factor. Therefore $\varepsilon = 0$ and the desired property follows.

1.0.3 Weak abstractness

Due to the interaction between denotations and continuations the full abstractness problem raised by Robin Milner [16][17] seems to be particularly difficult in continuation semantics. As far as we know, there is no full abstractness result published for a concurrent language designed semantically with continuations, although there are several papers that employ the technique of continuation in the semantic description of concurrent languages [3, 2, 24, 10].

A denotational semantics is said to be fully abstract with respect to a corresponding operational semantics if it is *correct* and *complete* (formal definitions are included in Section 2).[1] It is easy to show that a denotational model designed with CSC is correct. The correctness condition for the language $\mathcal{L}$ investigated in this paper was demonstrated in [27]. In general, the completeness condition is more difficult to establish. We do not know whether a fully abstract domain for CSC exists. The completeness condition can be stated as follows. If the denotations of two statements are different then there exists a syntactic context (a program expression with 'holes') in which the operational behavior of the two statements is also different.

In continuation semantics the completeness condition of full abstractness can be expressed as follows. If there exists a continuation for which the denotations of two statements are different then there exist a syntactic context in which the operational behavior of the two statements is also different (because the denotations of programs are functions that take continuations as arguments).

In this paper we propose a criterion - that we call *weak abstractness* - which can be used to asses the optimality of a denotational model designed with continuations. A denotational model designed with continuations is weakly abstract if it is *correct* and *weakly complete*.

[1] For the full abstractness problem we adopt the terminology in chapter 17 of [3].

The correctness condition is inherited from the full abstractness criterion (it is described formally in Section 2.2). Only the completeness condition is different. In words, the weak completeness condition can be described as follows. If there exists a *denotable* continuation for which the denotations of two statements are different then there exist a syntactic context in which the operational behavior of the two statements is also different.

The weak abstractness criterion is similar to the full abstractness criterion. It is weaker only in the sense that the completeness condition has to be verified merely for the class of denotable continuatuions. We do not know whether a fully abstract domain of CSC exists, but we show it is possible to identify a domain of denotable continuations (constructed by metric completion from the class of denotable continuations) which satisfies the desired completeness condition.

In Section 7 we show that the denotational semantics of $\mathcal{L}$ given in this paper is both correct and weakly abstract with respect to a corresponding operational semantics, hence the denotational semantics of $\mathcal{L}$ is weakly abstract.

1.0.4 Structure of the paper

Section 2 presents some theoretical preliminaries based mainly on the monograph [3]. In Section 3 we present the syntax of $\mathcal{L}$. In Section 4 we define a denotational semantics designed with metric spaces and CSC for the asynchronous language $\mathcal{L}$. In Section 5 we investigate the concurrency laws that are satisfied by our denotational model. The notion of weak abstractness is introduced in Section 6. In Section 7 we present the weak abstractness result for $\mathcal{L}$. Section 8 provides some conclusions together with possible directions of future research.

2 Preliminaries

The notation $(x \in)X$ introduces the set X with typical element x ranging over X. For any set X we denote by $|X|$ the *cardinal number* of X. Let $f \in X \to Y$ be a function with domain X and codomain Y. Let S be a subset of X, $S \subseteq X$. $f \restriction S$ denotes the function f restricted to the domain S, i.e. $f \restriction S : S \to Y$, $f \restriction S(x) = f(x), \forall x \in S$. If $f \in X \to Y$, we use the notation $(f \mid x \mapsto y)$ to represent the function $(f \mid x \mapsto y) : X \to Y$ defined by:

$$(f \mid x \mapsto y)(x') = \begin{cases} y & \text{if } x' = x \\ f(x') & \text{if } x' \neq x \end{cases}$$

$(f \mid x_1 \mapsto y_1, \cdots, x_n \mapsto y_n)$ is an abbreviation for $((f \mid x_1 \mapsto y_1) \cdots \mid x_n \mapsto y_n)$. If $f : X \to X$ and $f(x) = x$ we call x a *fixed point* of f. When this fixed point is unique (see Theorem 2.6) we write $x = fix(f)$.

The denotational semantics given in this paper is built within the mathematical framework of *1-bounded complete metric spaces*.

Definition 2.1 *A* metric space *is a pair* (M, d) *with* M *a non-empty set and* d *a mapping* $d : M \times M \to [0, 1]$ *which satisfies*

(a) $\forall x, y \in M\,[d(x, y) = 0 \Leftrightarrow x = y]$

(b) $\forall x, y \in M\,[d(x, y) = d(y, x)]$

(c) $\forall x, y, z \in M\,[d(x, y) \leq d(x, z) + d(z, y)]$

d *is called a* metric *or* distance. *Property (c) is the so-called* triangle inequality. *We call* (M, d) *an* ultra-metric space *if* M *is as before and the mapping* d *satisfies (a),(b) and the following stronger version of property (c):*

(c') $\forall x, y, z \in M : d(x, y) \leq max\{d(x, z), d(z, y)\}$

Definition 2.2 *Let* (M, d) *be a metric space and let* $(x_i)_i$ *be a sequence in* M.[2]

(a) $(x_i)_i$ *is a* convergent sequence *with* limit $x \in M$ *whenever*

$$\forall \varepsilon > 0 \exists n \in \mathbb{N} \forall i \geq n\,[d(x_i, x) \leq \varepsilon]$$

We say that $(x_i)_i$ converges to x *and we write* $\lim_i x_i = x$.

(b) $(x_i)_i$ *is a* Cauchy sequence *whenever*

$$\forall \varepsilon > 0 \exists n \in \mathbb{N} \forall i, j \geq n\,[d(x_i, x_j) \leq \varepsilon]$$

(b) The metric space (M, d) *is called* complete *whenever each Cauchy sequence converges to an element of* M.

Remarks 2.3 *Each convergent sequence is a Cauchy sequence. In a complete metric space each Cauchy sequence is convergent.*

2.0.5 Examples.

The following metrics are frequently used in semantic applications.

1. If $(x, y \in) X$ is any nonempty set, one can define the *discrete metric* on X as follows: $d(x, y) = 0$ if $x = y$, and $d(x, y) = 1$ if $x \neq y$. (X, d) is a complete ultrametric space.

2. A central idea in metric semantics is to state that two computations have distance 2^{-n} whenever the first difference in their behaviors appears after n computation steps. Let $(a \in) A$ be a nonempty set, and let $(x, y \in) A^\infty = A^* \cup A^\omega$, where $A^*(A^\omega)$ is the set of all finite (infinite) sequences over A. One can define a metric over A^∞ as follows: $d(x, y) = 2^{-\sup\{\, n \,|\, x(n) = y(n) \,\}}$, where $x(n)$ denotes the prefix of x of length n, in case $length(x) \geq n$, and x otherwise (by convention, $2^{-\infty} = 0$). d is a Baire-like metric. (A^∞, d) is a complete ultrametric space.

[2] A sequence in M, usually denoted by $(x_i)_i$, is a function $f : \mathbb{N} \to M$, with $x_i = f(i)$.

Definition 2.4 *Let $(M_1, d_1), (M_2, d_2)$ be metric spaces.*

(a) We say that (M_1, d_1) and (M_2, d_2) are isometric *if there exists a bijection $f : M_1 \to M_2$ such that $\forall x, y \in M_1\, [d_2(f(x), f(y)) = d_1(x, y)]$. We then write $M_1 \cong M_2$.*

(b) A function $f : M_1 \to M_2$ is called continuous *whenever, for each convergent sequence $(x_i)_i$ with $\lim_i x_i = x$ we also have that $\lim_i f(x_i) = f(x)$.*

(c) A function $f{:}M_1 \to M_2$ is a contraction *if $\exists c \in \mathbb{R}$, $0 \leq c < 1$, $\forall x, y \in M_1$ $[d_2(f(x), f(x)) \leq c \cdot d_1(x, y)]$. f is called* nonexpansive *when $c = 1$. We denote the set of all nonexpansive functions from M_1 to M_2 by $M_1 \stackrel{1}{\to} M_2$.*

Remark 2.5 *Each nonexpansive function is continuous.*

The Banach fixed point theorem is at the core of metric semantics.

Theorem 2.6 (Banach) *Let (M, d) be a complete metric space. Each contraction $f : M \to M$ has a* unique *fixed point.*

Definition 2.7 *Let $(M, d), (M', d')$ be (ultra) metric spaces. On the space $(x \in) M$, $(f \in) M \to M'$ (the function space), $(x, x') \in M \times M'$ (the Cartesian product), $u, v \in M + M'$ (the disjoint union of M and M', which can be defined by: $M + M' = (\{1\} \times M) \cup (\{2\} \times M')$), and $U, V \in \mathcal{P}(M)$ (the power set of M) one can define the following metrics:*

(a) $d_{\frac{1}{2} \cdot M} : M \times M \to [0, 1], \quad d_{\frac{1}{2} \cdot M}(x_1, x_2) = \frac{1}{2} \cdot d_M(x_1, x_2)$

(b) $d_{M \to M'} : (M \to M') \times (M \to M') \to [0, 1]$

$$d_{M \to M'}(f_1, f_2) = \sup_{x \in M} d_{M'}(f_1(x), f_2(x))$$

(c) $d_{M \times M'} : (M \times M') \times (M \times M') \to [0, 1]$

$$d_{M \times M'}((x_1, x_1'), (x_2, x_2')) = max\{d_M(x_1, x_2), d_{M'}(x_1', x_2')\}$$

(d) $d_{M+M'} : (M + M') \times (M + M') \to [0, 1]$

$$d_{M+M'}(u, v) = \begin{cases} d(u, v) & \text{if } u, v \in \{1\} \times M \\ d'(u, v) & \text{if } u, v \in \{2\} \times M' \\ 1 & \text{otherwise} \end{cases}$$

(e) $d_H : \mathcal{P}(M) \times \mathcal{P}(M) \to [0, 1]$*:*

$$d_H(U, V) = max\{\sup_{u \in U} d(u, V), \sup_{v \in V} d(v, U)\}$$

where $d(u, W) = \inf_{w \in W} d(u, w)$ and by convention $\sup \emptyset = 0$, $\inf \emptyset = 1$ (d_H is the Hausdorff metric*).*

We use the abbreviations $\mathcal{P}_{co}(\cdot)$ ($\mathcal{P}_{nco}(\cdot)$) to denote the power set of *compact* (*non-empty and compact*) subsets of '$\cdot$' (we assume known the notion of a *compact* set; see, e.g., [3]). Also, we often suppress the metrics part in domain definitions, and write, e.g., $\frac{1}{2}\cdot M$ instead of $(M, d_{\frac{1}{2}\cdot M})$.

Remark 2.8 *Let $(M,d),(M',d'),d_{\frac{1}{2}\cdot M}$, $d_{M\to M'}, d_{M\times M'}$, $d_{M+M'}$ and d_H be as in Definition 2.7. In case d, d' are ultrametrics, then so are $d_{\frac{1}{2}\cdot M}, d_{M\to M'}, d_{M\times M'}$, $d_{M+M'}$ and d_H. Moreover, if $(M,d),(M',d')$ are complete then $\frac{1}{2}\cdot M$, $M\to M'$, $M \xrightarrow{1} M'$, $M\times M'$, $M+M'$, $\mathcal{P}_{co}(M)$ and $\mathcal{P}_{nco}(M)$ (with the metrics defined above) are also complete metric spaces [3].*

We also use the abbreviation $\mathcal{P}_{fin}(\cdot)$ to denote the power sets of *finite* subsets of '$\cdot$'. In general, the construct $\mathcal{P}_{fin}(\cdot)$ does not give rise to a complete space. In our study, we use it to create a structure that we endow with the discrete metric. Any set endowed with the discrete metric is a complete ultrametric space.

2.1 Metric completion

Let $(M,d),(M',d')$ be metric spaces. We use the notation $(M,d)\lhd(M',d')$, or simply $M\lhd M'$, whenever (M is a subspace of M', i.e) $M\subseteq M'$ and $d'\restriction M = d$ (the restriction of d' to M coincides with d).

Definition 2.9 *Let (M,d) be a metric space. A* metric completion *of (M,d) is complete metric space (M',d') such that $M\lhd M'$ and for each $x\in M'$ we have: $x=\lim_i x_i$, with $x_i\in M, \forall i\in\mathbb{N}$ (the limit is taken with respect to d').*

Propozition 2.10 *Each metric space has a completion which is unique up to isometry.*

Remark 2.11 *Let (M,d) be a complete metric space and let X be a subset of M, $X\subseteq M$. We use the notation $co(X|M)$ to represent the set*

$$co(X|M) \stackrel{not.}{=} \{x \mid x\in M, x=\lim_i x_i, \forall i\in\mathbb{N}: x_i\in X,$$
$$(x_i)_i \text{ is a Cauchy sequence in } X\}$$

where the limits are taken with respect to d (as (M,d) is complete $\lim_i x_i\in M$). If we endow X with $d_X = d\restriction X$ and $co(X|M)$ with $d_{co(X|M)} = d\restriction co(X|M)$, then $(co(X|M), d_{co(X|M)})$ is a metric completion of (X,d_X).

Proof: It is easy to see that $X\lhd co(X|M)$ ($X\subseteq co(X|M)$ because $\forall x'\in X: x' = \lim_i x_i$, with $x_i = x', \forall i\in\mathbb{N}$, hence $x'\in co(X|M)$) and $co(X|M)\lhd M$. We prove that $(co(X|M), d_{co(X|M)})$ is a complete metric space. Let $(x_i)_i$ be a Cauchy sequence in $co(X|M)$ ($x_i\in co(X|M), \forall i\in\mathbb{N}$). As (M,d) is complete $x=\lim_i x_i\in M$. By the definition of $co(X|M)$, for each $i\in\mathbb{N}: x_i = \lim_j x_{ij}$, where $(x_{ij})_j$ is a Cauchy sequence in X ($x_{ij}\in X$, $\forall i,j\in\mathbb{N}$).

We show that $(x_{ii})_i$ is a Cauchy sequence (in (X, d_X)) and $\lim_i x_{ii} = x$. Let $\varepsilon > 0$. $\exists n \in \mathbb{N} \forall i, j \geq n : d(x_{ii}, x_i) \leq \frac{1}{3}, d(x_i, x_j) \leq \frac{1}{3}$, $d(x_j, x_{jj}) \leq \frac{1}{3}$ and $d(x_i, x) \leq \frac{1}{3}$. Hence, $(x_{ii})_i$ is a Cauchy in X sequence because $d_X(x_{ii}, x_{jj}) = d(x_{ii}, x_{jj}) \leq d(x_{ii}, x_i) + d(x_i, x_j) + d(x_j, x_{jj}) \leq \varepsilon$. Also, $\lim_i x_{ii} = x$ because $d(x_{ii}, x) \leq d(x_{ii}, x_i) + d(x_i, x) \leq \varepsilon$. Therefore $x \in co(X|M)$, which implies that the space $co(X|M)$ is complete.

By the definition of $co(X|M)$, for each $x \in co(X|M), x = \lim_i x_i$, with $x_i \in X, \forall i \in \mathbb{N}$; the limits can be taken with respect to $d_{co(X|M)} = d\restriction co(X|M)$. We conclude that $(co(X|M), d_{co(X|M)})$ is a metric completion of (X, d_X). □

2.2 Full abstractness

The full abstractness problem was raised by Robin Milner [16, 17]. A denotational semantics $\mathcal{D} : L \to \mathbf{D}$ is said to be *fully abstract* with respect to a (corresponding) operational semantics $\mathcal{O} : L \to \mathbf{O}$ if it is correct and complete.[3] $\mathcal{D}$ is said to be *correct* with respect to $\mathcal{O}$ if the following condition holds:

$$\forall x_1, x_2 \in L[\mathcal{D}(x_1) = \mathcal{D}(x_2) \Rightarrow \forall S[\mathcal{O}(S(x_1)) = \mathcal{O}(S(x_2))]]$$

where S is a syntactic context for L. Intuitively, a syntactic context is a language construct with 'holes'. The class of syntactic context for the asynchronous language $\mathcal{L}$ is given in Definition 5.6. When S is a syntactic context for L, we denote by $S(x)$ the result of 'plugging in' x for all occurrences of the 'hole' $(\cdot)$ in S.

The denotational model is said to be *complete* if whenever the denotations of two statements are different the difference can also be observed operationally in some syntactic context. Formally, $\mathcal{D}$ is complete with respect to $\mathcal{O}$ if:

$$\forall x_1, x_2 \in L[\mathcal{D}(x_1) \neq \mathcal{D}(x_2) \Rightarrow \exists S[\mathcal{O}(S(x_1)) \neq \mathcal{O}(S(x_2))]]$$

3 Syntax of $\mathcal{L}$

The syntax of $\mathcal{L}$ is given in BNF in Definition 3.1. The basic components are a set $(a \in) Act$ of *atomic actions* and a set $(y \in) Y$ of *recursion variables*. There is a special symbol $\delta \in Act$, whose behavior is explained below. ;, + and $\|$ are operators for sequential, nondeterministic and parallel composition, respectively. $\|$ is also called a *merge* operator, and $\lfloor\!\lfloor$ is the *left merge* operator.

Definition 3.1 *(Syntax of $\mathcal{L}$)*

(a) (Statements) $x(\in X) ::= a \mid y \mid x + x \mid x; x \mid x \lfloor\!\lfloor x \mid x \parallel x$

(b) (Guarded statements) $g(\in G) ::= a \mid g + g \mid g; x \mid g \lfloor\!\lfloor x \mid g \| g$

[3] In the full abstraction problem we adopt the terminology from chapter 17 of [3].

(d) (Declarations) $(D \in)Decl = Y \to G$

(e) (Programs) $(\rho \in)\mathcal{L} = Decl \times X$

The meaning of atomic actions is defined by an interpretation function $I : Act \to \Sigma \to (\{\uparrow\} \cup \Sigma)$, where $(\sigma \in)\Sigma$ is a set of *states*. If $I(a)(\sigma) = \uparrow$ the action a cannot proceed in state σ; its execution is *suspended*. When all processes are suspended *deadlock* occurs. Notice that $I(\delta)(\sigma) = \uparrow, \forall \sigma \in \Sigma$, i.e. the action δ suspends in all states. $\mathcal{L}$ incorporates the mechanism of *asynchronous communication* studied in [8]. As explained in [8], this form of asynchronous communication can be encountered in concurrent constraint programming, and also in other languages like dataflow or asynchronous CSP.

We employ an approach to recursion based on *declarations* and *guarded statements* [3]. In a guarded statement each recursive call is preceded by at least one elementary action, which guarantees the fact that the semantic operators are contracting functions in the present metric setting. For the sake of brevity (and without loss of generality) in what follows we assume a fixed declaration $D \in Decl$, and all considerations in any given argument refer to this fixed D.

For inductive proofs we introduce a complexity measure that decreases upon recursive calls. c_x is well-defined due to our restriction to guarded recursion.

Definition 3.2 *(Complexity measure) The function* $c_x : X \to \mathbb{N}$ *is given by*

$$\begin{aligned} c_x(a) &= 1 \\ c_x(y) &= 1 + c_x(D(y)) \\ c_x(x_1 \ \mathsf{op} \ x_2) &= 1 + c_x(x_1) \quad \mathsf{op} \in \{;, \between\} \\ c_x(x_1 \ \mathsf{op} \ x_2) &= 1 + max\{c_x(x_1), c_x(x_2)\} \quad \mathsf{op} \in \{+, \|\} \end{aligned}$$

4 Continuation Semantics for $\mathcal{L}$

In this section we introduce a denotational semantics designed with metric spaces and CSC for $\mathcal{L}$. It is based on the denotational model given in section 4 of [27]. In order to obtain the weak abstractness result given in Section 7 of this paper we define a notion of *consistent continuation*, and we define the behavior such that any $\mathcal{L}$ program blocks immediately if it is evaluated with respect to a nonconsistent continuation.

The consistency condition is simple and it represents an invariant property that is preserved by the denotational semantics. The initial continuation is consistent and each equation defining the denotational semantics preserves the consistency condition.

4.1 Structure of continuations

In the continuation semantics for concurrency (CSC) approach [27] the structure of continuations represents the main design tool. As explained in [27], in order to model the general

combination of sequential and parallel composition in $\mathcal{L}$ continuations can be structured as trees of computations with active elements at the leaves. For example, when the denotation of a program fragment $(x_1 \parallel x_2); x_3$ is computed, the denotations of x_1 and x_2 become leaves in such a tree and the denotation of x_3 becomes an inner node. This behavior is inspired by the concept of a *cactus stack* [7], a stack with multiple tops that can be active concurrently. Following [27],[12],[11] we model the domain of trees of computations with the aid of a (partially ordered) set of *identifiers* A. $(\alpha \in)A$ is the set of all finite, possibly empty (ϵ), sequences over $\{1, 2\}$ and $\alpha \leq \alpha'$ iff α is a prefix of α'.

Definition 4.1

(a) Let $(\alpha \in)A = \{1, 2\}^*$ *be a set of* identifiers*, equipped with the following partial ordering:* $\alpha \leq \alpha'$ *iff* $\alpha' = \alpha \cdot i_1 \cdots i_n$ *for* $i_1, \cdots, i_n \in \{1, 2\}, n \geq 0$.

(b) We define a function $max : \mathcal{P}(A) \to \mathcal{P}(A)$ *by:*

$$max(\pi) = \{\alpha \mid \alpha \text{ is a maximal element of } (\pi, \leq_\pi)\}$$

$\pi \in \mathcal{P}(A)$ *and* $\leq_\pi$ *is the restriction of* $\leq$ *to the subset* π *of* A.

The construct $(A, \leq)$ can be used to represent tree-like structures. For example, let $\pi = \{\alpha, \alpha \cdot 1, \alpha \cdot 2, \alpha \cdot 1 \cdot 1, \alpha \cdot 1 \cdot 2, \alpha \cdot 2 \cdot 1, \alpha \cdot 2 \cdot 2\}$. The maximal elements of $(\pi, \leq_\pi)$ are the *leaves* of the tree: $max(\pi) = \{\alpha \cdot 1 \cdot 1, \alpha \cdot 1 \cdot 2, \alpha \cdot 2 \cdot 1, \alpha \cdot 2 \cdot 2\}$. In this paper we use the symbol '$\cdot$' as a concatenation operator over sequences.

Let $(x \in)\mathbf{X}$ be a metric domain, i.e. a complete metric space. Following [12, 11], we also use the following notation:

$$\{\!|\mathbf{X}|\!\} \stackrel{not.}{=} \mathcal{P}_{fin}(A) \times (A \to \mathbf{X})$$

Let $\alpha \in A, \pi \in \mathcal{P}_{fin}(A), (\pi, \varphi) \in \{\!|\mathbf{X}|\!\}$, and $\varphi \in A \to \mathbf{X}$. We introduce operators $id(\cdot) : \{\!|\mathbf{X}|\!\} \to \mathcal{P}_{fin}(A)$, $(\cdot)(\cdot) : (\{\!|\mathbf{X}|\!\} \times A) \to \mathbf{X}$, $(\cdot) \setminus (\cdot) : (\{\!|\mathbf{X}|\!\} \times \mathcal{P}_{fin}(A)) \to \{\!|\mathbf{X}|\!\}$ and $[\cdot \mid \cdot \mapsto \cdot] : (\{\!|\mathbf{X}|\!\} \times A \times \mathbf{X}) \to \{\!|\mathbf{X}|\!\}$, defined by:

$$\begin{array}{rcl} id(\pi, \varphi) & = & \pi \\ (\pi, \varphi)(\alpha) & = & \varphi(\alpha) \\ (\pi, \varphi) \setminus \pi' & = & (\pi \setminus \pi', \varphi) \\ [(\pi, \varphi) \mid \alpha \mapsto x] & = & (\pi \cup \{\alpha\}, (\varphi \mid \alpha \mapsto x)) \end{array}$$

Sometimes, we use the notation $[(\pi, \varphi) \mid \alpha_1 \mapsto x_1 \mid \cdots \mid \alpha_n \mapsto x_n]$ as an abbreviation for $[\cdots[(\pi, \varphi) \mid \alpha_1 \mapsto x_1] \cdots \mid \alpha_n \mapsto x_n]$.

We also define $\nu : (A \times \{\!|\mathbf{X}|\!\}) \to \mathit{Bool}$ by

$$\nu(\alpha, (\pi, \varphi)) \;=\; (\alpha \notin \pi) \text{ and } (\alpha \in (max(\{\alpha\} \cup \pi)))$$

We treat (π, φ) as a 'function' with finite graph $\{(\alpha, \varphi(\alpha)) \mid \alpha \in \pi\}$, thus ignoring the behaviour of φ for any $\alpha \notin \pi$ (π is the 'domain' of (π, φ)). We use this mathematical

structure to represent finite partially ordered *bags* (or multisets)[4] of computations. The set A is used to distinguish between multiple occurrences of a computation in such a bag. We endow both sets A and $\mathcal{P}_{fin}(A)$ with discrete metrics; every set with a discrete metric is a complete ultrametric space. By using the composite metrics given in Definition 2.6 $\{\!|\mathbf{X}|\!\}$ becomes also a metric domain.

Remark 4.2 *Let* $\mathbf{X}$ *be a metric domain, and let* $((\alpha,(\pi,\varphi)),(\alpha',(\pi',\varphi')) \in \{\!|\mathbf{X}|\!\}$. *If* $\alpha \neq \alpha'$ *or* $\pi \neq \pi'$ *then* $d((\alpha,(\pi,\varphi)),(\alpha',(\pi',\varphi'))) = 1$.

The operators behave as follows. $id(\pi,\varphi)$ returns the collection of identifiers for the valid computations contained in the bag (π,φ), $(\pi,\varphi)(\alpha)$ returns the computation with identifier α, $(\pi,\varphi) \setminus \pi'$ removes the computations with identifiers in π', and $[(\pi,\varphi) \mid \alpha \mapsto x]$ replaces the computation with identifier α. The predicate $\nu(\alpha,(\pi,\varphi))$ is true iff the identifier α is a leaf of the tree $(\pi',\leq_{\pi'})$, with $\pi' = \{\alpha\} \cup \pi$, and $\alpha \notin \pi$. We use the operator ν to specify an invariant property for continuations; further explanations are provided after we introduce the domain of continuations.

By a slight abuse, we use the same notations (including the operators $id(\cdot)$, $(\cdot)(\cdot)$, $(\cdot) \setminus (\cdot)$, $[\cdot \mid \cdot \mapsto \cdot]$ and $\nu(\cdot,\cdot)$) when $(x \in)X$ is an ordinary set: $\{\!|X|\!\} = \mathcal{P}_{fin}(A) \times (A \to X)$; in this case we do not endow $\{\!|X|\!\}$ with a metric.

4.2 Semantic domain

We design a continuation-based denotational semantics for $\mathcal{L}$. As a semantic universe for the final yield of our denotational model we employ a standard linear-time domain $(p \in)\mathbf{P} = \mathcal{P}_{nco}(\Sigma^* \cup \Sigma^* \cdot \{\delta\} \cup \Sigma^\omega)$. Here $\Sigma^*(\Sigma^\omega)$ denotes the collection of all finite (infinite) sequences over Σ. An element of $\Sigma^* \cdot \{\delta\}$ is a finite sequence terminated with the symbol δ, which denotes *deadlock*. Also, we use the symbol ϵ to represent the empty sequence. We view $(q \in)\Sigma^* \cup \Sigma^* \cdot \{\delta\} \cup \Sigma^\omega$ as a complete ultrametric space by endowing it with the Baire metric (see section 2). The type of the denotational semantics $[\![\cdot]\!]$ for $\mathcal{L}$ is $X \to \mathbf{D}$, where:

$$\begin{array}{rcl} \mathbf{D} & \cong & \mathbf{Cont} \stackrel{1}{\to} \Sigma \to \mathbf{P} \\ (\gamma \in)\mathbf{Cont} & = & A \times \mathbf{Kont} \\ (\kappa \in)\mathbf{Kont} & = & \{\!|\tfrac{1}{2} \cdot \mathbf{D}|\!\} \end{array}$$

Notice that the recursive occurence of $\mathbf{D}$ is preceded by '$\frac{1}{2}\cdot$ ' and it is placed in the left-hand side of a nonexpansive function space. According to [1] the above domain equation has a solution, which is unique up to isometry. The solution for $\mathbf{D}$ is obtained as a complete ultrametric space.

As in [12, 11], we call an element of **Kont** a *closed continuation* and an element of **Cont** an *open continuation*. A continuation is a representation of what remains to be computed from the program [26]. A closed continuation $\kappa \in \mathbf{Kont}$ is a self-contained

[4] We avoid using the notion of a *partially ordered multiset* which is a more refined structure – see [5], or chapter 16 of [3].

structure of computations. An open continuation behaves like an evaluation context [13] for the denotational mapping $[\![\cdot]\!]$.

4.3 Denotational semantics

The denotational function $[\![\cdot]\!]$ is defined in 4.4 with the aid of a mapping kc, which is called a *scheduler*. Since the definition of the denotational mapping $[\![\cdot]\!]$ is very similar to the ones given in [27][12][11], we adopt a more terse style of presentation. Only the consistency condition given in Definition 4.3 is specific of the denotational semantics given in this paper. No semantic property is affected by this design decision. Moreover, the proofs are are trivial for nonconsistent continuations and similar to the proofs provided in citeCT10,CT13 for consistent continuations. In addition, the denotational model given in this paper is both *weakly abstract*, a notion to be introduced formally in Section 6.

We use the notation $\sigma \cdot p = \{\sigma \cdot q \mid q \in p\}$, for any $\sigma \in \Sigma$ and $p \in \mathbf{P}$. Also, the semantics of nondeterministic choice in $\mathcal{L}$ is given by the operator $+ : (\mathbf{P} \times \mathbf{P}) \to \mathbf{P}$. The definition reflects that $p_1 + p_2$ blocks only if both p_1 and p_2 block (the symbol δ denotes deadlock). It is easy to check that $+$ is well-defined, nonexpansive, associative, commutative and idempotent.

$$p_1 + p_2 = \{q \mid q \in p_1 \cup p_2, q \neq \delta\} \cup \{\delta \mid \delta \in p_1 \cap p_2\}$$

Definition 4.3 *Let $(\alpha, \kappa) \in \mathbf{Cont}$ be an open continuation. We say (α, κ) is* consistent *if $\nu(\alpha, \kappa)$. Also, we say that (α, κ) is* inconsistent *if $\neg\nu(\alpha, \kappa)$.*

In the Definition 4.4 we use the following convention:

$$\Phi(\phi)(\alpha, \kappa)(\sigma) =_c p$$

is an abbreviation for:

$$\Phi(\phi)(\alpha, \kappa)(\sigma) = \begin{cases} p & \mathsf{if}\ \nu(\alpha, \kappa) \\ \{\delta\} & \mathsf{otherwise} \end{cases}$$

for $\Phi : (X \to \mathbf{D}) \to (X \to \mathbf{D})$, $\phi \in X \to \mathbf{D}$, $(\alpha, \kappa) \in \mathbf{Cont}$, $\sigma \in \Sigma$, $p \in \mathbf{P}$.

Definition 4.4 *(Denotational semantics for $\mathcal{L}$)*

(a) Let $kc : \mathbf{Kont} \to \Sigma \to \mathbf{P}$ be given by:

$$kc(\kappa)(\sigma) \;=\; \mathsf{if}\ (id(\kappa) = \emptyset)\ \mathsf{then}\ \{\epsilon\}\ \mathsf{else}\ +_{\alpha \in max(id(\kappa))}\ \kappa(\alpha)(\alpha, \kappa \setminus \{\alpha\})(\sigma)$$

(b) We define $\Phi : (X \to \mathbf{D}) \to (X \to \mathbf{D})$*, for* $\phi \in X \to \mathbf{D}$*, by:*

$$\begin{array}{rcl}
\Phi(\phi)(a)(\alpha,\kappa)(\sigma) & =_c & \mathsf{if}\ (I(a)(\sigma) = \uparrow)\ \mathsf{then}\ \{\delta\} \\
 & & \mathsf{else}\ I(a)(\sigma) \cdot kc(\kappa)(I(a)(\sigma)) \\
\Phi(\phi)(y)(\alpha,\kappa)(\sigma) & =_c & \Phi(\phi)(D(y))(\alpha,\kappa)(\sigma) \\
\Phi(\phi)(x_1 + x_2)(\alpha,\kappa)(\sigma) & =_c & \Phi(\phi)(x_1)(\alpha,\kappa)(\sigma) \ + \ \Phi(\phi)(x_2)(\alpha,\kappa)(\sigma) \\
\Phi(\phi)(x_1; x_2)(\alpha,\kappa)(\sigma) & =_c & \Phi(\phi)(x_1)(\alpha \cdot 1, [\kappa \mid \alpha \mapsto \phi(x_2)])(\sigma) \\
\Phi(\phi)(x_1 \between x_2)(\alpha,\kappa)(\sigma) & =_c & \Phi(\phi)(x_1)(\alpha \cdot 1, [\kappa \mid \alpha \cdot 2 \mapsto \phi(x_2)])(\sigma) \\
\Phi(\phi)(x_1 \parallel x_2)(\alpha,\kappa)(\sigma) & =_c & \Phi(\phi)(x_1)(\alpha \cdot 1, [\kappa \mid \alpha \cdot 2 \mapsto \phi(x_2)])(\sigma) \ + \\
 & & \Phi(\phi)(x_2)(\alpha \cdot 1, [\kappa \mid \alpha \cdot 2 \mapsto \phi(x_1)])(\sigma)
\end{array}$$

(c) We put $[\![\cdot]\!] = fix(\Phi)$*. Let* $\kappa_0 = (\emptyset, \lambda\alpha.[\![\delta]\!])$*, and* $\alpha_0 = \epsilon$*.* $(\alpha_0, \kappa_0) \in \mathbf{Cont}$ *is the initial open continuation. Notice that* $kc(\kappa_0)(\sigma) = \{\epsilon\}$*, for any* $\sigma \in \Sigma$*. We also define* $\mathcal{D}[\![\cdot]\!] : X \to \Sigma \to \mathbf{P}$ *by:*

$$\mathcal{D}[\![x]\!] = [\![x]\!](\alpha_0, \kappa_0)$$

An open continuation (α, κ) is consistent iff $\nu(\alpha, \kappa)$. The consistency condition is an invariant of the denotational semantics. The initial continuation is consistent. Also, each equation defining the denotational semantics preserves the consistency condition. Notice that, if (α, κ) is an inconsistent open continuation then $[\![x]\!](\alpha, \kappa)(\sigma) = \{\delta\}$, for any $x \in X$, $\sigma \in \Sigma$.

Definition 4.4 is justified by Lemmas 4.5 and 4.6, whose proofs are omitted. Similar Lemmas are given in [27, 28]. See, e.g., the proofs of Lemmas 3.13 and 3.14 in [27].

Lemma 4.5

(a) The mapping kc (given in Definition 4.4) is well-defined.

(b) $\forall \kappa_1, \kappa_2 \in \mathbf{Kont} : d(kc(\kappa_1), kc(\kappa_2)) \leq 2 \cdot d(\kappa_1, \kappa_2)$

Lemma 4.6 *For all* $\phi \in (X \to \mathbf{D}), x \in X, (\alpha, \kappa) \in \mathbf{Cont}, \sigma \in \Sigma$*:*

(a) $\Phi(\phi)(x)(\alpha, \kappa)(\sigma) \in \mathbf{P}$ *(it is well defined),*

(b) $\Phi(\phi)(x)$ *is nonexpansive (in* (α, κ)*), and*

(c) Φ *is* $\frac{1}{2}$ *- contractive (in* ϕ*).*

5 Concurrency Laws

For the asynchronous language $\mathcal{L}$ that we study in this paper, in [12, 11] we showed that the semantic operators designed with CSC satisfy laws that are usually included in concurrency theories, such as the associativity and commutativity of parallel composition. Moreover, we obtained a complete axiomatization of parallel composition inspired by classic process algebra theories [6]. In this section we repeat the results given in [12][11] adapted to the denotational semantics introduced in this paper. Actually all properties remain valid and all proofs are as similar to the ones provided in [12][11] if we restrict to the class of consistent continuations. Also, the proofs are trivial for nonconsistent continuations. In general, the semantic properties are only satisfied for the class of denotable continuations, a notion to be defined formally below. We introduce the auxiliary notion of a *resumption* and a notion of isomorphism over resumptions. A continuation may contain arbitrary values of the type **D**. We prove the desired properties for continuations that can be obtained as semantified versions of resumptions, i.e. for for continuations that contain only denotations of statements; such continuations we call *denotable continuations*. The evaluation with respect to denotable continuations represents an invariant property of the denotational semantics, because the initial continuation is empty and the denotational semantics adds to the continuation only denotations of statements.

Definition 5.1

(a) We define the set of closed resumptions $(k \in)KRes = \{\!|X|\!\}$,[5] *where X is the class of $\mathcal{L}$ statements. We define the set $CRes$ of* open resumptions *by $CRes = A \times KRes$. Let $(\alpha, k) \in CRes$ be an open resumption. We say that (α, k) is* consistent *if $\nu(\alpha, k)$. We say that (α, k) is* inconsistent *if $\neg\nu(\alpha, k)$.*

*(b) We define $[\![\cdot]\!] : KRes \to$ **Kont**, $[\![k]\!] = (id(k), \lambda\alpha.[\![k(\alpha)]\!])$.*

Remark 5.2 *Let $(\alpha, k) \in CRes$ be an open resumption. $\nu(\alpha, k) \Leftrightarrow \nu(\alpha, [\![k]\!])$, i.e. (α, k) is consistent iff $(\alpha, [\![k]\!])$ is consistent.*

A continuation that contains only computations denotable by program statements we call a *denotable continuation*. The class of closed denotable continuations is $(\kappa \in)Kont^{\mathcal{D}} = \{[\![k]\!] \mid k \in KRes\}$. The class of open denotable continuations is $((\alpha, \kappa) \in)Cont^{\mathcal{D}} = \{(\alpha, [\![k]\!]) \mid (\alpha, k) \in CRes\} = A \times Kont^{\mathcal{D}}$. Denotable continuations are the semantic counterpart of resumptions. Obviously, $[\![k]\!] \in$ **Kont** for any $k \in KRes$ and $(\alpha, [\![k]\!]) \in$ **Cont** for any $(\alpha, k) \in CRes$. Not all continuations are denotable, but in general the semantic properties can only be proved for denotable continuations.

Definition 5.3 *We say that two open resumptions $(\alpha_1, k_1), (\alpha_2, k_2) \in CRes$ are* isomorphic*, and write $(\alpha_1, k_1) \cong (\alpha_2, k_2)$, iff either of the following two conditions ((1) or (2)) is satisfied:*

[5] In this case the construct $\{\!| \cdot |\!\}$ is used to define an ordinary set; see the explanation given in the final part of subsection 4.1.

(1) $\neg\nu(\alpha_1, k_1)$ *and* $\neg\nu(\alpha_1, k_1)$ *(*(α_1, k_1) *and* (α_1, k_1) *are both inconsistent)*

(2) $\nu(\alpha_1, k_1)$ *and* $\nu(\alpha_1, k_1)$ *(*(α_1, k_1) *and* (α_1, k_1) *are both consistent) and there exists a bijection* $\mu : (\{\alpha_1\} \cup id(k_1)) \to (\{\alpha_2\} \cup id(k_2))$ *such that:*

(i) $\mu(\alpha_1) = \alpha_2$

(ii) $\mu(\alpha') \leq \mu(\alpha'') \Leftrightarrow \alpha' \leq \alpha'', \forall \alpha', \alpha'' \in id(\{\alpha_1\} \cup id(k_1))$

(iii) $k_2(\mu(\alpha')) = k_1(\alpha'), \quad \forall \alpha' \in id(k_1)$

Obviously, an inconsistent resumption cannot be isomorphic with a consistent resumption, and $\forall(\alpha, k) \in CRes : (\alpha, k) \cong (\alpha, k)$. It is easy to prove the following Lemma.

Lemma 5.4 $\forall k \in KRes, \alpha \in A, x \in X$: $[\![[k \mid \alpha \mapsto x]]\!] = [[\![k]\!] \mid \alpha \mapsto [\![x]\!]]$

Proposition 5.5 states that any two denotable continuations that correspond to isomorphic resumptions behave the same. The proof technique for such properties was introduced in [12][11]. The basic idea is to identify a computation invariant (as a relation between continuation structures) that is preserved by each computation step. In the case of Proposition 5.5 the relevant property that is preserved by the computation steps is the isomorphism of resumptions that correspond to denotable continuations. Next, the desired property follows by using an argument '$\varepsilon \leq \frac{1}{2} \cdot \varepsilon \Rightarrow \varepsilon = 0$'. which is standard in metric semantics. The proof of Proposition 5.5 is similar to the proof of Corollary 5.6 given in [11].

Propozition 5.5 *For all* $x \in X, (\alpha_1, k_1), (\alpha_2, k_2)$ $(\in CRes)$ *with* $(\alpha_1, k_1) \cong (\alpha_2, k_2)$*:*
$$[\![x]\!](\alpha_1, [\![k_1]\!]) = [\![x]\!](\alpha_2, [\![k_2]\!]).$$

In [12][11] we showed that continuations can be used to reason in a compositional manner upon the behavior of concurrent programs, based on a notion of syntactic context for the class of $\mathcal{L}$ statements.

Definition 5.6 *(Syntactic contexts for* $\mathcal{L}$*)*

$$S ::= (\cdot) \mid a \mid y \mid S; S \mid S + S \mid S \| S \mid S \between S$$

We denote by $S(x)$ *the result of substituting* x *for all occurrences of* $(\cdot)$ *in* S*. Formally, this substitution can be defined inductively:*
$(\cdot)(x) = x$, $a(x) = a$, $y(x) = y$, $(S \ \mathsf{op} \ S')(x) = S(x) \ \mathsf{op} \ S'(x)$, *where* $\mathsf{op} \in \{;, +, \|, \between\}$.

Theorem 5.7 states the main properties of the semantic operators in the continuation semantics of $\mathcal{L}$. Some properties hold for all continuations and can be proved by simple manipulations of the semantic equations. Other properties only hold for denotable continuations and require more complex proofs based on the identification of computation invariants (as relations between continuation structures) in combination with the '$\varepsilon \leq \frac{1}{2} \cdot \varepsilon \Rightarrow \varepsilon = 0$' argument. The evaluation with respect to denotable and consistent continuations represents

an invariant of the denotational semantics. The proof of Theorem 5.7 is ommited because it is similar to the proof of Theorem 5.12 given in [11].

We write $x \simeq \overline{x}$ $(x, \overline{x} \in X)$ to express that $[\![S(x)]\!](\alpha, [\![k]\!]) = [\![S(\overline{x})]\!](\overline{\alpha}, [\![\overline{k}]\!])$ for all $\mathcal{L}$ syntactic contexts S and for all denotable continuations corresponding to isomorphic resumptions $(\alpha, k) \cong (\overline{\alpha}, \overline{k})$ $(\in CRes)$. As it is well-known, the properties stated by Theorem 5.7 provide a finite axiomatization for the parallel composition operator $\|$; see, e.g., [4].

Theorem 5.7 *For all $x, x_1, x_2, x_3 \in X$:*

(a)	$x_1 + x_2$	$\simeq$	$x_2 + x_1$	(*commutativity of* $+$)
(b)	$(x_1 + x_2) + x_3$	$\simeq$	$x_1 + (x_2 + x_3)$	(*associativity of* $+$)
(c)	$x + x$	$\simeq$	x	(*idempotency of* $+$)
(d)	$(x_1 + x_2); x_3$	$\simeq$	$x_1; x_3 + x_2; x_3$	(*right distributivity of* ; over $+$)
(e)	$x_1; (x_2; x_3)$	$\simeq$	$(x_1; x_2); x_3$	(*associativity of* ;)
(f)	$x + \delta$	$\simeq$	x	
(g)	δx	$\simeq$	δ	
(h)	$x_1 \parallel x_2$	$\simeq$	$x_1 \between x_2 + x_2 \between x_1$	
(i)	$a \between x$	$\simeq$	$a; x$	
(j)	$(a; x_1) \between x_2$	$\simeq$	$a; (x_1 \parallel x_2)$	
(k)	$(x_1 + x_2) \between x_3$	$\simeq$	$x_1 \between x_3 + x_2 \between x_3$	(*right distributivity of* $\between$ over $+$)
(l)	$x_1 \parallel x_2$	$\simeq$	$x_2 \parallel x_1$	(*commutativity of* $\|\|$)
(m)	$x_1 \| (x_2 \parallel x_3)$	$\simeq$	$(x_1 \| x_2) \parallel x_3$	(*associativity of* $\|\|$)

Remarks 5.8

(a) There is no language construct $x \in X$ such that $\phi_\epsilon = [\![x]\!]$, where $\phi_\epsilon = \lambda(\alpha, \kappa).\lambda\sigma\,.\,\{\epsilon\}$. Indeed, $\phi_\epsilon \in \mathbf{D}$ is nonexpansive in (α, κ), but $d([\![x]\!], \phi_\epsilon) = 1$, hence $[\![x]\!] \neq \phi_\epsilon)$, $\forall x \in X$. $\epsilon \notin [\![x]\!](\alpha, \kappa)(\sigma), \forall x \in X, (\alpha, \kappa) \in \mathbf{Cont}, \sigma \in \Sigma$, as can be verified easily by induction on $c_x(x)$. If $\epsilon \notin p$ then $d(p, \{\epsilon\}) = 1$, $\forall p \in \mathbf{P}$ (because if $q \neq \epsilon$ then $d(q, \epsilon) = 1, \forall q \in \mathbf{Q}$). Therefore, $d([\![x]\!], \phi_\epsilon) = \sup_{(\alpha,\kappa)\in\mathbf{Cont},\sigma\in\Sigma} d([\![x]\!](\alpha, \kappa)(\sigma), \{\epsilon\}) = 1$.

(b) Not all continuations are denotable. *Let $\gamma_\epsilon = (1, [\kappa_0 \mid \epsilon \mapsto \phi_\epsilon])$, $\gamma_\epsilon \in \mathbf{Cont}$. The open continuation γ_ϵ is* not *denotable because it contains the computation $\phi_\epsilon = \lambda(\alpha, \kappa).\lambda\sigma\,.\,\{\epsilon\}$ which is not denotable by any language construct. In fact, $d(\gamma, \gamma_\epsilon) = \frac{1}{2}$, for any $\gamma \in Cont^{\mathcal{D}} = \{(\alpha, [\![k]\!]) \mid (\alpha, k) \in CRes\}$.*

For further explanations and examples of calculations based on the denotational semantics $[\![\cdot]\!]$ the reader may consult [12][11]. When evaluation is restricted to the class of consistent continuations the denotational semantics provided in this paper and the denotational semantics given in [12][11] behave the same.

5.1 The domain of denotable continuations

In the sequel we continue our investigation based on the the class of closed denotable continuations $(\kappa \in) Kont^{\mathcal{D}} = \{[\![k]\!] \mid k \in KRes\}$ and the class of open denotable continuations $((\alpha,\kappa) \in) Cont^{\mathcal{D}} = \{(\alpha, [\![k]\!]) \mid (\alpha, k) \in CRes\}$ (introduced after Definition 5.1) . We do not know whether $Kont^{\mathcal{D}}$ and $Cont^{\mathcal{D}}$ are metric domains, i.e. a complete metric spaces. However, in the CSC approach computations are nonexpansive continuous functions, $\mathbf{D} \cong \mathbf{Cont} \stackrel{1}{\to} \Sigma \to \mathbf{P}$. As an immediate implication, in CSC computations are also continuous functions [3]. Based on the continuity of the computations in the sequel we study semantic properties in a metric completion of the $Cont^{\mathcal{D}}$ which is a subspace of **Cont**.

Obviously, $(\alpha, [\![k]\!]) \in \mathbf{Cont}$, for any $(\alpha, k) \in CRes$. According to Remark 2.11, we can construct the metric completion of $Cont^{\mathcal{D}}$ with respect to **Cont**. Let $((\alpha,\kappa) \in)\mathbf{Cont}^{\mathcal{D}} = co(Cont^{\mathcal{D}}|\mathbf{Cont})$. We explained in (the proof of) Remark 2.11 that $Cont^{\mathcal{D}} \lhd \mathbf{Cont}^{\mathcal{D}}$ and $\mathbf{Cont}^{\mathcal{D}} \lhd \mathbf{Cont}$. We call $\mathbf{Cont}^{\mathcal{D}}$ the *domain of (open) denotable continuations*.

Definition 5.9 *We say that two open denotable continuations* $(\alpha,\kappa), (\overline{\alpha},\overline{\kappa}) \in \mathbf{Cont}^{\mathcal{D}}$ *are* isomorphic*, and we write* $(\alpha,\kappa) \cong (\overline{\alpha},\overline{\kappa})$*, iff there exist sequences* $(\alpha_i, k_i)_i$*,* $(\overline{\alpha}_i, \overline{k}_i)_i$*, (with* $(\alpha_i, k_i), (\overline{\alpha}_i, \overline{k}_i) \in CRes, \forall i \in \mathbb{N}$*) such that:* $(\alpha,\kappa) = \lim_i(\alpha_i, [\![k_i]\!])$*,* $(\overline{\alpha},\overline{\kappa}) = \lim_i(\overline{\alpha}_i, [\![\overline{k}_i]\!])$*, and* $(\alpha_i, k_i) \cong (\overline{\alpha}, \overline{k}_i), \forall i \in \mathbb{N}$.[6]

By Remark 4.2, if $(\alpha,\kappa) = \lim_i(\alpha_i, [\![k_i]\!])$ then there exists $n \in \mathbb{N}$ such that $\alpha = \alpha_i$ and $id(\kappa) = id([\![k_i]\!])$, for any $i \geq n$.

We write $x \sim \overline{x}$ $(x, \overline{x} \in X)$ to express that $[\![S(x)]\!](\alpha,\kappa) = [\![S(\overline{x})]\!](\overline{\alpha},\overline{\kappa})$ for all $\mathcal{L}$ syntactic contexts S and for all isomorphic open denotable continuations $(\alpha,\kappa) \cong (\overline{\alpha},\overline{\kappa})$, with $(\alpha,\kappa), (\overline{\alpha},\overline{\kappa}) \in \mathbf{Cont}^{\mathcal{D}}$.

Remark 5.10 $x \simeq \overline{x} \Rightarrow x \sim \overline{x}$*, for all* $x, \overline{x} \in X$.

Proof: Let $(\alpha,\kappa), (\overline{\alpha},\overline{\kappa}) \in \mathbf{Cont}^{\mathcal{D}}$ be two isomorphic open denotable continuations, $(\alpha,\kappa) \cong (\overline{\alpha},\overline{\kappa})$, and let S be an $\mathcal{L}$ syntactic context. There exist sequences of open resumptions $(\alpha_i, k_i)_i$, $(\overline{\alpha}_i, \overline{k}_i)_i$ $((\alpha_i, k_i), (\overline{\alpha}_i, \overline{k}_i) \in CRes, \forall i \in \mathbb{N})$ such that: $(\alpha,\kappa) = \lim_i(\alpha_i, [\![k_i]\!])$, $(\overline{\alpha},\overline{\kappa}) = \lim_i(\overline{\alpha}_i, [\![\overline{k}_i]\!])$ and $(\alpha_i, k_i) \cong (\overline{\alpha}_i, \overline{k}_i)$, $\forall i \in \mathbb{N}$. As $x \simeq \overline{x}$, $[\![S(x)]\!](\alpha_i, [\![k_i]\!]) = [\![S(\overline{x})]\!](\overline{\alpha}_i, [\![\overline{k}_i]\!])$, for all $i \in \mathbb{N}$. Therefore, $\forall \varepsilon > 0 \exists n \in \mathbb{N} \forall i \geq n$:

$d([\![S(x)]\!](\alpha,\kappa), [\![S(\overline{x})]\!](\overline{\alpha},\overline{\kappa}))$ [strong triangle inequality applied twice]

$\leq max\{d([\![S(x)]\!](\alpha,\kappa), [\![S(x)]\!](\alpha_i, [\![k_i]\!])),$

$d([\![S(x)]\!](\alpha_i, [\![k_i]\!]), [\![S(\overline{x})]\!](\overline{\alpha}_i, [\![\overline{k}_i]\!])),$

$d([\![S(\overline{x})]\!](\overline{\alpha}_i, [\![\overline{k}_i]\!]), [\![S(\overline{x})]\!](\overline{\alpha},\overline{\kappa}))\}$

$[[\![S(x)]\!], [\![S(\overline{x})]\!]$ are nonexpansive mappings]

[6]The isomorphism relation between open resumptions is defined in Definition 5.3.

$\leq max\{d((\alpha,\kappa),(\alpha_i,[\![k_i]\!])),d((\overline{\alpha},\overline{\kappa}),(\overline{\alpha}_i,[\![\overline{k}_i]\!]))\} \leq \varepsilon$

Hence, $d([\![S(x)]\!](\alpha,\kappa),[\![S(\overline{x})]\!](\overline{\alpha},[\![\overline{k}]\!])) = 0$, i.e. $[\![S(x)]\!](\alpha,\kappa) = [\![S(\overline{x})]\!](\overline{\alpha},[\![\overline{k}]\!])$. In this proof the $\mathcal{L}$ syntactic context S and the pair of isomorphic denotable continuations $(\alpha,\kappa) \cong (\overline{\alpha},\overline{\kappa})$ are arbitrarily selected. Therefore, $x \sim \overline{x}$. □

As a consequence of Remark 5.10, all concurrency laws stated in Theorem 5.7 for the class of denotable continuations also hold in the whole domain of denotable continuations $\mathbf{Cont}^{\mathcal{D}}$.

Remarks 5.11

(a) Let $(\alpha,\kappa),(\overline{\alpha},\overline{\kappa}) \in \mathbf{Cont}^{\mathcal{D}}$ be open denotable continuations. It is easy to check that I $\Leftrightarrow$ II.

I. $(\alpha,\kappa) \cong (\overline{\alpha},\overline{\kappa})$ ((α,κ) and $(\overline{\alpha},\overline{\kappa})$ are isomorphic)

II. Either of the following two conditions ((1) or (2)) is satisfied:

(1) $\neg\nu(\alpha,\kappa)$ and $\neg\nu(\overline{\alpha},\overline{\kappa})$, or

(2) $\nu(\alpha,\kappa)$ and $\nu(\overline{\alpha},\overline{\kappa})$ and there exists a bijection $\mu : (\{\alpha\} \cup id(\kappa)) \to (\{\overline{\alpha}\} \cup id(\overline{\kappa}))$ such that:

(i) $\mu(\alpha) = \overline{\alpha}$

(ii) $\mu(\alpha') \leq \mu(\alpha'') \Leftrightarrow \alpha' \leq \alpha'', \forall \alpha', \alpha'' \in id(\{\alpha\} \cup id(\kappa))$

(iii) $\overline{\kappa}(\mu(\alpha')) = \kappa(\alpha'), \quad \forall \alpha' \in id(\kappa)$

(c) Let $\gamma_\epsilon (\in \mathbf{Cont})$, $\gamma_\epsilon = (1, [\kappa_0 \mid \epsilon \mapsto \phi_\epsilon])$, where $\phi_\epsilon = \lambda(\alpha,\kappa).\lambda\sigma.\{\epsilon\}$. According to Remark 5.8(b), $d(\gamma,\gamma_\epsilon) = \frac{1}{2}$, for any open denotable continuation $\gamma \in Cont^{\mathcal{D}}$. As the distance between γ_ϵ and any denotable continuation $\gamma \in Cont^{\mathcal{D}}$ is $\geq \frac{1}{2}$, we cannot find a Cauchy sequence $(\gamma_i)_i$, with $\gamma_i \in Cont^{\mathcal{D}}, \forall i \in \mathbb{N}$, such that $\lim_i \gamma_i = \gamma_\epsilon$. The implication is that $\gamma_\epsilon \in \mathbf{Cont}$ but $\gamma_\epsilon \notin \mathbf{Cont}^{\mathcal{D}}$, which means that ($\mathbf{Cont}^{\mathcal{D}} \lhd \mathbf{Cont}$ but) $\mathbf{Cont}^{\mathcal{D}} \neq \mathbf{Cont}$.

6 Weak abstractness for CSC

In the CSC approach the correctness of a denotational semantics with respect to a corresponding operational semantics is easy to establish [27] but we do not know whether a fully abstract domain of CSC exists. In fact, we are not aware of any fully abstract domain for a concurrent language designed semantically with continuations, although the technique of continuations is used in various papers that present denotational models for concurrent languages [3], [2], [24], [10].

In continuation semantics the *completeness* condition of full abstractness can be expressed as follows: if there exists a continuation for which the denotations of two statements are different then there exists a syntactic context in which the operational behaviors of the two statements are also different. Let $\mathcal{D} : L \to \mathbf{D}$ be a denotational semantics for a language $(x \in)L$, where $\mathbf{D} = \mathbf{Cont} \to \mathbf{F}$ is a domain of denotations given as functions from a domain $(\gamma \in)\mathbf{Cont}$ of continuations to a domain $\mathbf{F}$ of final program answers. The syntactic contexts of L are ranged over by the variable symbol S. The completeness condition of full abstractness can be expressed formally by using the notation employed in subsection 2.2 as follows:

$$\forall x_1, x_2 \in L[\,(\exists \gamma \in \mathbf{Cont}[\mathcal{D}(x_1)\gamma \neq \mathcal{D}(x_2)\gamma]) \Rightarrow (\exists S[\mathcal{O}(S(x_1)) \neq \mathcal{O}(S(x_2))])\,]$$

We cannot prove this general completeness condition for CSC. But there is a weaker condition - that we call *weak completeness* - which can be established for CSC. This condition is similar to the completeness condition of full abstractness, but it is weaker in the sense that it has to be established mereley for the class of denotable continuations.

In Definition 6.1 we introduce a class of complete metric spaces which is large enough to express arbitrary continuation domains for CSC. Based on this class of metric domains, in Definition 6.2 we generalize the notion of a denotable continuation. The weak abstractness condition is introduced as a general criterion in the context of CSC in Definition 6.3.

Definition 6.1 *In the CSC approach a metric domain of denotations is defined as the solution (which must be unique, up to isometry) of an equation* $\mathbf{D} \cong \mathbf{\Gamma} \stackrel{1}{\to} \mathbf{F}$*, where:*

- $\mathbf{\Gamma}$ *is a (typical) element of the class* $\mathcal{CONT}$ *of* domains for CSC*, given by:*

  $$\mathbf{\Gamma} ::= \tfrac{1}{2} \cdot \mathbf{D} \mid \mathbf{M} \to \mathbf{\Gamma} \mid \mathbf{M} \times \mathbf{\Gamma} \mid \mathbf{M} + \mathbf{\Gamma} \mid \mathbf{\Gamma} \times \mathbf{\Gamma} \mid \mathbf{\Gamma} + \mathbf{\Gamma}$$

  $\mathbf{M}$ *is an arbitrary set* $(m \in)M$ *endowed with the discrete metric. The constructions* $\frac{1}{2} \cdot (\cdot), \stackrel{1}{\to}, \to, \times$ *and* $+$ *are given in Definition 2.7 (*$\mathbf{M}$ *is endowed with the discrete metric, hence any function in* $\mathbf{M} \to \mathbf{\Gamma}$ *is nonexpansive). Notice that* $\frac{1}{2} \cdot \mathbf{D}$ *occurs at least once in any* $\mathbf{\Gamma} \in \mathcal{CONT}$*.*

- $\mathbf{F}$ *is a metric domain designed such that the equation* $\mathbf{D} \cong \mathbf{\Gamma} \stackrel{1}{\to} \mathbf{F}$ *has a unique solution (up to isometry) according to the general theory presented in [1]. If the domain* $\mathbf{F}$ *does not depend on* $\mathbf{D}$ *the existence and the uniqueness of the solution of the equation* $\mathbf{D} \cong \mathbf{\Gamma} \stackrel{1}{\to} \mathbf{F}$ *is guaranteed [1].*

In CSC continuations are *multisets* of computations (intuitively, computations are processes that can be executed in parallel) rather than plain sets of computations. Multisets can be represented as functions of the type $\mathbf{M} \to \mathbf{\Gamma}$, where $\mathbf{M}$ is used as an *index* domain, whose elements are needed to distinguish between multiple occurences of computations. Hence, powerdomain constructions (e.g. the compact powerdomain construction $\mathcal{P}_{co}(\mathbf{\Gamma})$) are lacking from the list of constructions for continuations given in Definition 6.1.

We define a general notion of denotable continuation in the context of the class of metric domains for CSC introduced in Definition 6.1.

Definition 6.2 *(Denotable continuations)*

(a) Let $(x \in)X$ *be a fixed set. We define a class* $\mathcal{RES}(X)$ *of* sets of resumptions *for* X*, with typical element* R*, as follows:*

$$R ::= X \mid M \to R \mid M \times R \mid M + R \mid R \times R \mid R + R$$

$(m \in)M$ *is an arbitrary set.* $M \to R$ *is the set of functions from* M *to* R*.* $M \times R$ *and* $M + R$ *are the cartesian product and disjoint union of* M *and* C*, respectively. Similarly,* $R_1 \times R_2$ *is the cartesian product of* R_1 *and* R_2*, and* $R_1 + R_2$ *is the disjoint union of* R_1 *and* R_2*.* [7]

(b) For any language $(x \in)L$ *and CSC domain* $\mathbf{\Gamma}$ *we can construct a corresponding set of resumptions. The correspondence we give by a homomorphism* $res^L(\cdot) : \mathcal{CONT} \to \mathcal{RES}(L)$ *defined inductively on the structure of* $\mathbf{\Gamma}$*.*

$$\begin{aligned}
res^L(\frac{1}{2} \cdot \mathbf{D}) &= L \\
res^L(\mathbf{M} \to \mathbf{\Gamma}) &= M \to res^L(\mathbf{\Gamma}) \\
res^L(\mathbf{M} \times \mathbf{\Gamma}) &= M \times res^L(\mathbf{\Gamma}) \\
res^L(\mathbf{M} + \mathbf{\Gamma}) &= M + res^L(\mathbf{\Gamma}) \\
res^L(\mathbf{\Gamma}_1 \times \mathbf{\Gamma}_2) &= res^L(\mathbf{\Gamma}_1) \times res^L(\mathbf{\Gamma}_2) \\
res^L(\mathbf{\Gamma}_1 + \mathbf{\Gamma}_2) &= res^L(\mathbf{\Gamma}_1) + res^L(\mathbf{\Gamma}_2)
\end{aligned}$$

$res^L(\cdot)$ *maps a complete metric space to a plain set.* $\mathbf{M}$ *is a metric space, given by an arbitrary set* $(m \in)M$ *endowed with the discrete metric.*

(c) Let $(x \in)L$ *be a language and let* $\mathcal{D} : L \to \mathbf{D}$ *be a denotational semantics of* L *designed with CSC, where* $\mathbf{D} \cong \mathbf{\Gamma} \xrightarrow{1} \mathbf{F}$ *is as in Definition 6.1.* $(\gamma \in)\Gamma^{\mathcal{D}} = \{[\![c]\!]^L_{\mathbf{\Gamma}} \mid c \in res^L(\mathbf{\Gamma})\}$

[7] $R_1 + R_2 = (\{1\} \times R_1) \cup (\{2\} \times R_2)$, as in Definition 2.7 (and similarly for $M + R$), although here R_1, R_2 and $R_1 + R_2$ $(M + R)$ are plain sets, rather than metric spaces.

is the class of denotable continuations *for* $\mathcal{D}$*, where for any* $\mathbf{\Gamma} \in \mathcal{CONT}$*,* $[\![\cdot]\!]^L_{\mathbf{\Gamma}}$: $res^L(\mathbf{\Gamma}) \to \mathbf{\Gamma}$ *is given by*

$$\begin{aligned}
[\![x]\!]^L_{\frac{1}{2}\cdot\mathbf{D}} &= \mathcal{D}(x)\\
[\![\lambda m\,.\,c]\!]^L_{\mathbf{M}\to\mathbf{\Gamma}} &= \lambda m\,.\,[\![c]\!]^L_{\mathbf{\Gamma}}\\
[\![(m,c)]\!]^L_{\mathbf{M}\times\mathbf{\Gamma}} &= (m,[\![c]\!]^L_{\mathbf{\Gamma}})\\
[\![(i,c)]\!]^L_{\mathbf{M}+\mathbf{\Gamma}_2} &= \begin{cases} (1,m) & \text{if } i=1\\ (2,[\![c]\!]^L_{\mathbf{\Gamma}_2}) & \text{if } i=2\end{cases}\\
[\![(c_1,c_2)]\!]^L_{\mathbf{\Gamma}_1\times\mathbf{\Gamma}_2} &= ([\![c_1]\!]^L_{\mathbf{\Gamma}_1},[\![c_2]\!]^L_{\mathbf{\Gamma}_2})\\
[\![(i,c)]\!]^L_{\mathbf{\Gamma}_1+\mathbf{\Gamma}_2} &= \begin{cases} (1,[\![c]\!]^L_{\mathbf{\Gamma}_1}) & \text{if } i=1\\ (2,[\![c]\!]^L_{\mathbf{\Gamma}_2}) & \text{if } i=2\end{cases}
\end{aligned}$$

Clearly, any element of $\Gamma^{\mathcal{D}}$ *is also an element of* $\mathbf{\Gamma}$*. We have* $\Gamma^{\mathcal{D}} \lhd \mathbf{\Gamma}$*.*

(d) Let $(\gamma \in)\mathbf{\Gamma}^{\mathcal{D}} = co(\Gamma^{\mathcal{D}}|\mathbf{\Gamma})$ *be the metric completion of* $\Gamma^{\mathcal{D}}$ *with respect to* $\mathbf{\Gamma}$ *constructed as in Remark 2.11. We call* $\mathbf{\Gamma}^{\mathcal{D}}$ *the (metric)* domain of denotable continuations *for* $\mathcal{D}$*.*

In Definition 6.3 we introduce the weak abstractness criterion in the context of CSC. Lemma 6.4 states that the weak completeness property (introduced in Definition 6.3(b)) holds for all denotable continuations in $\mathbf{\Gamma}^{\mathcal{D}}$ whenever it holds for all denotable continuations in $\Gamma^{\mathcal{D}}$. But in general (and also in the particular case of the asynchronous language $\mathcal{L}$ that we study in this paper), we do not know whether the space $\Gamma^{\mathcal{D}}$ is complete or not (where it is, $\Gamma^{\mathcal{D}} = \mathbf{\Gamma}^{\mathcal{D}}$). Recall that $\Gamma^{\mathcal{D}} \lhd \mathbf{\Gamma}^{\mathcal{D}}$ and $\mathbf{\Gamma}^{\mathcal{D}} \lhd \mathbf{\Gamma}$ (see Remark 2.11). Also, in general, $\mathbf{\Gamma}^{\mathcal{D}} \neq \mathbf{\Gamma}$ (see Remark 5.11(b)).

Definition 6.3 *(Weak abstractness for CSC) Let* $(x \in)L$ *be a language and let* $\mathcal{D} : L \to \mathbf{D}$ *be a denotational semantics of* L *designed with CSC, where* $\mathbf{D} \cong \mathbf{\Gamma} \stackrel{1}{\to} \mathbf{F}$ *is as in Definition 6.1. Let also* $\mathcal{O} : L \to \mathbf{O}$ *be an operational semantics of* L *and* S *a typical element of the class of syntactic contexts for* L*.*

(a) $\mathcal{D}$ *is* correct *with respect to* $\mathcal{O}$ *iff*

$$\forall x_1, x_2 \in L[\mathcal{D}(x_1) = \mathcal{D}(x_2) \Rightarrow \forall S[\mathcal{O}(S(x_1)) = \mathcal{O}(S(x_2))]]$$

(b) Let $(\gamma \in)\mathbf{\Gamma}^{\mathcal{D}}$ *be the domain of denotable continuations for* $\mathcal{D}$*, as defined in 6.2(d). We say that* $\mathcal{D}$ *is* weakly complete *with respect to* $\mathcal{O}$ *iff*

$$\forall x_1, x_2 \in L[(\exists \gamma \in \mathbf{\Gamma}^{\mathcal{D}}[\mathcal{D}(x_1)\gamma \neq \mathcal{D}(x_2)\gamma]) \Rightarrow (\exists S[\mathcal{O}(S(x_1)) \neq \mathcal{O}(S(x_2))])]$$

(c) We say that $\mathcal{D}$ *is* weakly abstract *with respect to* $\mathcal{O}$ *iff* $\mathcal{D}$ *is correct and weakly complete with respect to* $\mathcal{O}$*.*

In words, the weak completeness condition for CSC can be expressed as follows. If there exists a *denotable* continuation for which the denotations of two statements are different then there exists a syntactic context in which the operational behavior of the two statements is also different. Denotable continuations are the denotational counterpart of the syntactic contexts.

Lemma 6.4 *Let* $(x \in)L$ *be a language and let* $\mathcal{D} : L \to \mathbf{D}$ *be a denotational semantics of* L *designed with CSC, where* $\mathbf{D} \cong \mathbf{\Gamma} \stackrel{1}{\to} \mathbf{F}$ *is as in Definition 6.1. Let also* $\mathcal{O} : L \to \mathbf{O}$ *be an operational semantics of* L *and* S *a typical element of the class of syntactic contexts for* L*. The denotational model* $\mathcal{D}$ *is weakly complete with respect to* $\mathcal{O}$ *iff*

$$\forall x_1, x_2 \in L[(\exists \gamma \in \Gamma^{\mathcal{D}}[\mathcal{D}(x_1)\gamma \neq \mathcal{D}(x_2)\gamma]) \Rightarrow (\exists S[\mathcal{O}(S(x_1)) \neq \mathcal{O}(S(x_2))])]$$

where $\Gamma^{\mathcal{D}}$ *is the class of denotable continuations for* $\mathcal{D}$*, as defined in 6.2(c).*

Proof: Let $(\gamma \in)\mathbf{\Gamma}^{\mathcal{D}}$ be the domain of denotable continuations for $\mathcal{D}$ (defined in 6.2(d)). And assume $\mathcal{D}(x_1)\gamma \neq \mathcal{D}(x_2)\gamma$, where $\gamma \in \mathbf{\Gamma}^{\mathcal{D}}$, i.e. $\gamma = \lim_i \gamma_i$, with $\gamma_i \in \Gamma^{\mathcal{D}}$, $\forall i \in \mathbb{N}$. The proof given below works for arbitrary metric domains, not just for ultrametric spaces (we use the triangle inequality, rather than the strong tringle inequality). For any $i \in \mathbb{N}$:

$d(\mathcal{D}(x_1)\gamma, \mathcal{D}(x_2)\gamma)$ [triangle inequality applied twice]

$\leq d(\mathcal{D}(x_1)\gamma, \mathcal{D}(x_1)\gamma_i) + d(\mathcal{D}(x_1)\gamma_i, \mathcal{D}(x_2)\gamma_i) + d(\mathcal{D}(x_2)\gamma_i, \mathcal{D}(x_2)\gamma)$

$[\mathcal{D}(x_1), \mathcal{D}(x_2) \in \mathbf{\Gamma} \stackrel{1}{\to} \mathbf{F}]$

$\leq 2 \cdot d(\gamma, \gamma_i) + d(\mathcal{D}(x_1)\gamma_i, \mathcal{D}(x_2)\gamma_i)$

As $\gamma = \lim_i \gamma_i$, $\forall \varepsilon > 0 \exists n \in \mathbb{N} \forall i > n : d(\gamma, \gamma_i) \leq \varepsilon$. Let $\varepsilon = \frac{1}{4} \cdot d(\mathcal{D}(x_1)\gamma, \mathcal{D}(x_2)\gamma)$. $d(\mathcal{D}(x_1)\gamma, \mathcal{D}(x_2)\gamma) > 0$ (and thus $\varepsilon > 0$) because $\mathcal{D}(x_1)\gamma \neq \mathcal{D}(x_2)\gamma$. Hence

$0 < \frac{1}{2} \cdot d(\mathcal{D}(x_1)\gamma, \mathcal{D}(x_2)\gamma) = d(\mathcal{D}(x_1)\gamma, \mathcal{D}(x_2)\gamma) - 2 \cdot \varepsilon$

$\leq d(\mathcal{D}(x_1)\gamma, \mathcal{D}(x_2)\gamma) - 2 \cdot d(\gamma, \gamma_i) \leq d(\mathcal{D}(x_1)\gamma_i, \mathcal{D}(x_2)\gamma_i)$

Therefore, $\exists i \in \mathbb{N} : \mathcal{D}(x_1)\gamma_i \neq \mathcal{D}(x_2)\gamma_i$. As $\gamma_i \in \Gamma^{\mathcal{D}}$, $\forall i \in \mathbb{N}$, we conclude that $\exists S[\mathcal{O}(S(x_1)) \neq \mathcal{O}(S(x_2))]$. □

7 Weak abstractness for the asynchronous language $\mathcal{L}$

In subsection 7.1 we present an operational semantics $\mathcal{O}[\![\cdot]\!]$ for $\mathcal{L}$ and we prove that the denotational model $[\![\cdot]\!]$ is *correct* with respect to $\mathcal{O}[\![\cdot]\!]$. The correctness proof is essentially taken from [27]. The distinction between the denotational semantics of $\mathcal{L}$ given in [27] and the denotational semantics $[\![\cdot]\!]$ given in this paper is given only by their behavior for the class of nonconsistent continuations. Notice that the operational semantics is only defined for consistent resumptions. In this paper we show that $[\![\cdot]\!]$ is also weakly complete and thus weakly abstract in the sense defined in section 6.

7.1 Operational semantics. Correctness of denotational semantics.

In Definition 7.2 we present the transition system for $\mathcal{L}$ (defined in the style of structured operational semantics [23]) on which it is based the operational semantics given in Definition 7.4. The configurations of the transition system are introduced in Definition 7.1.

Definition 7.1

(a) We define the class $((\alpha, k) \in) CRes'$ *of* consistent open resumptions*:* $CRes' = \{(\alpha, k) \mid \alpha \in A, k \in KRes, \nu(\alpha, k)\}$. *Obviously,* $CRes' \subseteq CRes$ *(see subsection 4.1 and Definition 5.1).*

(b) We also define the class $(t \in) Conf$ *of* configurations *of the transition relation for* $\mathcal{L}$*:* $Conf = (X \times CRes' \times \Sigma) \cup (KRes \times \Sigma)$.

In Definition 7.2 we use the following convention:

$t_1 \nearrow t_2$ is an abreviation for $\dfrac{t_2 \to t'}{t_1 \to t'}$

Definition 7.2 *(Transition system for* $\mathcal{L}$*) The transition relation for* $\mathcal{L}$ *is the smallest subset of* $Conf \times (KRes \times \Sigma)$ *satisfying the axioms and rules below:*

(A1) $(a, (\alpha, k), \sigma) \to (k, \sigma')$ if $I(a)(\sigma) = \sigma'$

(R2) $(y, (\alpha, k), \sigma) \nearrow (D(y), (\alpha, k), \sigma)$

(R3) $(x_1 + x_2, (\alpha, k), \sigma) \nearrow (x_1, (\alpha, k), \sigma)$

(R4) $(x_1 + x_2, (\alpha, k), \sigma) \nearrow (x_2, (\alpha, k), \sigma)$

(R5) $(x_1; x_2, (\alpha, k), \sigma) \nearrow (x_1, (\alpha \cdot 1, [k \mid \alpha \mapsto x_2]), \sigma)$

(R6) $(x_1 \parallel x_2, (\alpha, k), \sigma) \nearrow (x_1, (\alpha \cdot 1, [k \mid \alpha \cdot 2 \mapsto x_2]), \sigma)$

(R7) $(x_1 \parallel x_2, (\alpha, k), \sigma) \nearrow (x_2, (\alpha \cdot 1, [k \mid \alpha \cdot 2 \mapsto x_1]), \sigma)$

(R8) $(k, \sigma) \nearrow (k(\alpha), (\alpha, k \setminus \{\alpha\}), \sigma)$ $\forall \alpha \in max(id(k))$

The above transition system was introduced in section 4 of [27] (although in [27] the notation is slightly different). According to axiom (A1) an elementary action a can only be executed in a state σ where $I(a)(\sigma) \neq \uparrow$. Following the CSC approach, resumptions are trees of statements with active elements at the leaves (the maximal elements with respect to the partial order $\leq$ defined on A).

Definition 7.3

(a) Let $t \in Conf$. We use the notation $t \not\rightarrow$ to express the fact that t has no transitions, i.e. there is no t' such that $t \rightarrow t'$.

(b) If $t = (k, \sigma) \in KRes \times \Sigma$ and $id(k) = \emptyset$ we say that t terminates.

(c) If $t \not\rightarrow$ and t does not terminate we say that t blocks.

Let $t \in Conf$. It is always decidable whether $t \not\rightarrow$. Also, if $t \nearrow t'$ then $t' \in Conf$ and if $t \rightarrow t'$ then $t' \in KRes \times \Sigma$. We ommit the proofs of these simple properties because similar properties are established in [27].

Definition 7.4 *(Operational semantics $\mathcal{O}[\![\cdot]\!]$)*

(a) Let $(S \in) Sem_O = Conf \rightarrow \mathbf{P}$. We define $\Psi : (Conf \rightarrow \mathbf{P}) \rightarrow (Conf \rightarrow \mathbf{P})$

$$\Psi(S)(t) = \begin{cases} \{\epsilon\} & \text{if } t \text{ terminates} \\ \{\delta\} & \text{if } t \text{ blocks} \\ \bigcup\{\sigma \cdot S(k,\sigma) \mid t \rightarrow (k,\sigma)\} & \text{otherwise} \end{cases}$$

(b) Let $\alpha_0 = \epsilon$ and $k_0 = (\emptyset, \lambda\alpha\,.\,\delta)(\in KRes)$. Obviously, $(\alpha_0, k_0) \in CRes'$ and $(x, (\alpha_0, k_0), \sigma) \in Conf$. We put $\mathcal{O} = fix(\Psi)$. We define $\mathcal{O}[\![\cdot]\!] : X \rightarrow \Sigma \rightarrow \mathbf{P}$

$$\mathcal{O}[\![x]\!](\sigma) = \mathcal{O}(x, (\alpha_0, k_0), \sigma)$$

The transition system given in Definition 7.2 is finitely branching[8] and thus the operational semantics is compact.

We show that $\mathcal{D}[\![x]\!] = \mathcal{O}[\![x]\!]$, for any $x \in X$. For this purpose we introduce an auxiliary mapping $\mathcal{R} : Conf \rightarrow \mathbf{P}$, and we make fruitfull use of Banach's fixed point Theorem 2.6.

Definition 7.5 *Let $\mathcal{R} : Conf \rightarrow \mathbf{P}$ be given as follows. $\mathcal{R}(k, \sigma) = kc[\![k]\!](\sigma)$, for any $(k, \sigma) \in KRes \times \Sigma$. Also, $\mathcal{R}(x, (\alpha, k), \sigma) = [\![x]\!](\alpha, [\![k]\!])(\sigma)$, for any $(x, (\alpha, k), \sigma) \in X \times CRes' \times \Sigma$.*

Lemma 7.6 *Let $t \in Conf$.*

(a) If t terminates then $\mathcal{R}(t) = \{\epsilon\}$.

(b) If t blocks then $\mathcal{R}(t) = \{\delta\}$.

(c) If t does not block then $\delta \notin \mathcal{R}(t)$.

[8] For any $t \in Conf$ the set $\{(k, \sigma) \mid t \rightarrow (k, \sigma)\}$ is finite. One can check this in two steps. First, by induction on $c_x(x)$, for any $t = (x, (\alpha, k), \sigma) \in X \times CRes' \times \Sigma$. Next, for any $t = (k, \sigma) \in KRes \times \Sigma$, by using the fact that $max(id(k))$ is a finite set.

Lemma 7.7 $\mathcal{R} = fix(\Psi)$.

Proof: We prove that $\mathcal{R}(t) = \Psi(\mathcal{R})(t)$, $\forall t \in Conf$. If t terminates or t blocks then the result follows by Definition 7.4 and Lemma 7.6. If t has indeed transitions, by Lemma 7.6(c) $\delta \notin \mathcal{R}(t)$. We treat the case $t = (x, (\alpha, k), \sigma) \in X \times CRes' \times \Sigma$ and we proceed by induction on $c_x(x)$. We only consider one subcase $x = x_1 \parallel x_2$.

Let $t = (x_1 \parallel x_2, (\alpha, k), \sigma)$, $t_1 = (x_1, (\alpha \cdot 1, [k \mid \alpha \cdot 2 \mapsto x_2]), \sigma)$, $t_2 = (x_2, (\alpha \cdot 1, [k \mid \alpha \cdot 2 \mapsto x_1]), \sigma)$. It is easy to check that $\mathcal{R}(t) = \mathcal{R}(t_1) + \mathcal{R}(t_2)$. t does not block if at least one of t_1 and t_2 does not block. We only consider the case when both t_1 and t_2 have transitions, hence $t \to (k, \sigma) \Leftrightarrow (t_1 \to (k, \sigma) \vee t_2 \to (k, \sigma))$. By Lemma 7.6(c) $\delta \notin \mathcal{R}(t_1)$ and $\delta \notin \mathcal{R}(t_2)$. We have:

$$
\begin{aligned}
&\Psi(\mathcal{R})(t) \qquad [\text{Def. } \Psi] \\
&= \bigcup\{\sigma \cdot \mathcal{R}(k, \sigma) \mid t_1 \to (k, \sigma)\} \cup \bigcup\{\sigma \cdot \mathcal{R}(k, \sigma) \mid t_2 \to (k, \sigma)\} \\
&= \Psi(\mathcal{R})(t_1) \cup \Psi(\mathcal{R})(t_2) \qquad [\text{Induction hypothesis}] \\
&= \mathcal{R}(t_1) \cup \mathcal{R}(t_2) \qquad [\text{Def. '+', } \delta \notin \mathcal{R}(t_1),\ \delta \notin \mathcal{R}(t_2)] \\
&= \mathcal{R}(t_1) + \mathcal{R}(t_2) \\
&= \mathcal{R}(t)
\end{aligned}
$$

□

Theorem 7.8 $\mathcal{D}[\![x]\!] = \mathcal{O}[\![x]\!]$, *for any* $x \in X$.

Proof: Let $x \in X, \sigma \in \Sigma$. It is enough to show that $\mathcal{D}[\![x]\!](\sigma) = \mathcal{O}[\![x]\!](\sigma)$.

$$
\begin{aligned}
&\mathcal{D}[\![x]\!](\sigma) = [\![x]\!](\alpha_0, \kappa_0)(\sigma) \\
&= [\![x]\!](\alpha_0, [\![k_0]\!])(\sigma) \\
&= \mathcal{R}(x, (\alpha_0, k_0), \sigma) \qquad [\text{Definition 7.4, Lemma 7.7, Theorem 2.6}] \\
&= \mathcal{O}(x, (\alpha_0, k_0), \sigma) \\
&= \mathcal{O}[\![x]\!](\sigma)
\end{aligned}
$$

□

The function $[\![\cdot]\!]$ is defined according to the compositional principle which is characteristic of denotational semantics, but the auxiliary function $\mathcal{D}[\![x]\!] = [\![x]\!](\alpha_0, \kappa_0)$ introduced in Definition 4.4(c) is *not* a compositional mapping ($\mathcal{D}[\![\cdot]\!]$ is *not* a denotational semantics). The relation $\mathcal{D}[\![\cdot]\!] = \mathcal{O}[\![\cdot]\!]$ established in Theorem 7.8 is not a full abstractness result. However, we can establish the correctness of $[\![\cdot]\!]$ with respect to $\mathcal{O}[\![\cdot]\!]$. If $[\![x_1]\!] = [\![x_2]\!]$ then, by the compositionality of $[\![\cdot]\!]$, $[\![S(x_1)]\!] = [\![S(x_2)]\!]$, for any $\mathcal{L}$ syntactic context S. Hence, $\mathcal{O}[\![S(x_1)]\!] = \mathcal{D}[\![S(x_1)]\!] = \mathcal{D}[\![S(x_2)]\!] = \mathcal{O}[\![S(x_2)]\!]$ for any $\mathcal{L}$ syntactic context S. We conclude that $[\![\cdot]\!]$ is correct with respect to $\mathcal{O}[\![\cdot]\!]$.

7.2 Weak abstractness

If we expand the notation $\{\!|\cdot|\!\}$ in the definition of the domain of continuations (introduced at the begining of subsection 4.2) for $\mathcal{L}$ we get:

$$((\alpha,\kappa)\in)\mathbf{Cont} = A \times (\mathcal{P}_{fin}(A) \times (A \to \frac{1}{2}\cdot \mathbf{D}))$$

The sets A and $\mathcal{P}_{fin}(A)$ are endowed with discrete metrics. The definition is an instance of the general CSC framework presented in Section 6.

The corresponding set of open resumptions is:

$$((\alpha,k)\in)CRes = A \times (\mathcal{P}_{fin}(A) \times (A \to X))$$

Cont is an element of the class $\mathcal{CONT}$. $CRes = res^X(\mathbf{Cont})$ is an element of the class $\mathcal{RES}(X)$. It is easy to check that

$$[\![(\alpha,k)]\!]^X_{\mathbf{Cont}} = (\alpha, [\![k]\!])$$

where the mapping $[\![\cdot]\!]^X_{\mathbf{Cont}}$ is given in Definition 6.2 and the mapping $[\![\cdot]\!]$ is given in Definition 5.1. The class of open denotable continuations for $[\![\cdot]\!]$ is given by $Cont^{\mathcal{D}} = \{[\![(\alpha,k)]\!]^X_{\mathbf{Cont}} \mid (\alpha,k) \in res^X(\mathbf{Cont})\} = \{(\alpha,[\![k]\!]) \mid (\alpha,k)\in CRes\}$. Also, the domain of denotable continuations is $\mathbf{Cont}^{\mathcal{D}} = co(Cont^{\mathcal{D}}|\mathbf{Cont})$; see Remark 2.11. The denotational semantics $[\![\cdot]\!]$ of $\mathcal{L}$ defined in Section 4 is an instance of the general CSC framework presented in Section 6, hence Lemma 6.4 does hold for $[\![\cdot]\!]$.

The weak abstractness proof for $[\![\cdot]\!]$ concludes with Corollary 7.11. We use Lemma 7.9 to establish the completeness condition for $[\![\cdot]\!]$.

Lemma 7.9 *For any $x\in X$, $(\alpha,k)\in CRes$ there exists an $\mathcal{L}$ syntactic context S such that: $[\![x]\!](\alpha,[\![k]\!]) = \mathcal{D}[\![S(x)]\!]$ $(= [\![S(x)]\!](\alpha_0,\kappa_0)$, see Definition 4.4).*

Proof: By Definition 4.4, $\mathcal{D}[\![S(x)]\!] = [\![S(x)]\!](\alpha_0,\kappa_0) = [\![S(x)]\!](\alpha_0,[\![k_0]\!])$, where $\kappa_0 = (\emptyset,\lambda\alpha\,.\,[\![\delta]\!]) \in \mathbf{Kont}, k_0 = (\emptyset,\lambda\alpha\,.\,\delta) \in KRes$. If $\neg\nu(\alpha,[\![k]\!])$ we can take $S(\cdot)=\delta(\cdot)$, and the result is immediate, because $\mathcal{D}[\![\delta(x)]\!](\sigma) = \mathcal{D}[\![\delta]\!](\sigma) = \{\delta\}$. If $\nu(\alpha,[\![k]\!])$ the proof of Lemma 7.9 can proceed by induction on $|id(k)|$. In case $|id(k)|=0$ we have $(\alpha,k)\cong(\alpha_0,k_0)$. Therefore, by Corollary 5.5, $[\![x]\!](\alpha,[\![k]\!]) = [\![x]\!](\alpha_0,[\![k_0]\!]) = \mathcal{D}[\![x]\!] = \mathcal{D}[\![S(x)]\!]$, with $S=(\cdot)$.

In case $|id(k)|>0$, by Lemma .4(a) $\exists\alpha',\alpha''\in\{\alpha\}\cup id(k)$ such that either $seq(\alpha',\alpha'',\{\alpha\}\cup id(k))$ or $par(\alpha',\alpha'',\{\alpha\}\cup id(k))$ (see Definition .3 given in the appendix). [9] We only consider the case when $par(\alpha',\alpha'',\{\alpha\}\cup id(k))$. By Lemma .4(c) there are three subcases. We treat the subcase when $\alpha'\neq\alpha$, $\alpha''\neq\alpha$ and $(\alpha,k)\cong(\alpha,[k\setminus\{\alpha',\alpha''\} \mid \alpha'\cdot 1\mapsto k(\alpha') \mid \alpha'\cdot 2\mapsto k(\alpha'')])$. In this subcase α,α' and α'' are incomparable (because $\alpha'\neq\alpha,\alpha''\neq\alpha$, $\alpha'\neq\alpha''$ and $\alpha,\alpha',\alpha''\in max(\{\alpha\}\cup id(k))$).

[9] There are open resumptions (α,k) for which we can have both $seq(\alpha',\alpha'',\{\alpha\}\cup id(k))$ and $par(\alpha',\alpha'',\{\alpha\}\cup id(k))$. For such resumptions there may be several different $\mathcal{L}$ syntactic contexts $S_1,S_2,\cdots$ satisfying $[\![x]\!](\alpha,[\![k]\!]) = \mathcal{D}[\![S_i(x)]\!] = \mathcal{D}[\![S_j(x)]\!]$ (when $S_i\neq S_j$). The syntactic differences between S_i and S_j correspond to applications of concurrency laws such as the associativity of parallel composition.

$[\![x]\!](\alpha, [\![k]\!])$ [Corollary 5.5]

$= [\![x]\!](\alpha, [\![[k \setminus \{\alpha', \alpha''\} \mid \alpha' \cdot 1 \mapsto k(\alpha') \mid \alpha' \cdot 2 \mapsto k(\alpha'')]\!]\!])$ [Lemma 5.4]

$= [\![x]\!](\alpha, [[\![k \setminus \{\alpha', \alpha''\}]\!] \mid \alpha' \cdot 1 \mapsto [\![k(\alpha')]\!] \mid \alpha' \cdot 2 \mapsto [\![k(\alpha'')]\!]])$

[Lemma .2 $\nu(\alpha', k \setminus \{\alpha', \alpha''\})$, $\neg(\alpha \geq \alpha')$, $\neg(\alpha' \geq \alpha)$]

$= [\![x]\!](\alpha, [[\![k \setminus \{\alpha', \alpha''\}]\!] \mid \alpha' \mapsto [\![k(\alpha') \parallel k(\alpha'')]\!]])$ [Lemma 5.4]

$= [\![x]\!](\alpha, [\![[k \setminus \{\alpha', \alpha''\} \mid \alpha' \mapsto (k(\alpha') \parallel k(\alpha''))]\!]\!])$

As $|id(k)| > |id([k \setminus \{\alpha', \alpha''\} \mid \alpha' \mapsto (k(\alpha') \parallel k(\alpha''))])|$, by the induction hypothesis there exists an $\mathcal{L}$ syntactic context S such that:

$[\![x]\!](\alpha, [\![[k \setminus \{\alpha', \alpha''\} \mid \alpha' \mapsto (k(\alpha') \parallel k(\alpha''))]\!]\!])$

$= [\![S(x)]\!](\alpha_0, [\![k_0]\!]) = \mathcal{D}[\![S(x)]\!]$

□

Remarks 7.10 *Considering the way we constructed the $\mathcal{L}$ syntactic context S in the proof of Lemma 7.9 we notice the following.*

(a) S contains exactly one occurrence of the ('hole') symbol $(\cdot)$.

(b) The construction of the syntactic context S does not *depend on $x(\in X)$; it only depends on the resumption $(\alpha, k)(\in CRes)$. Hence, the proof of Lemma 7.9 shows that, for any $x, x' \in X$, $x \neq x'$, and for any resumption (α, k) we can always construct an $\mathcal{L}$ syntactic context S such that we have both $[\![x]\!](\alpha, [\![k]\!]) = \mathcal{D}[\![S(x)]\!]$ and $[\![x']\!](\alpha, [\![k]\!]) = \mathcal{D}[\![S(x')]\!]$ (the same S in the both equations).*

Corollary 7.11 *The denotational semantics $[\![\cdot]\!]$ of $\mathcal{L}$ is weakly abstract with respect to the operational semantics $\mathcal{O}[\![\cdot]\!]$.*

Proof: In subsection 7.1 we showed that $[\![\cdot]\!]$ is correct with respect to $\mathcal{O}[\![\cdot]\!]$. It remains to prove that $[\![\cdot]\!]$ is also weakly complete with respect to $\mathcal{O}[\![\cdot]\!]$. By Lemma 6.4, it suffices to show that

$$\forall x_1, x_2 \in X[(\exists(\alpha, \kappa) \in Cont^{\mathcal{D}}[[\![x_1]\!](\alpha, \kappa) \neq [\![x_2]\!](\alpha, \kappa)]) \Rightarrow$$
$$(\exists S[\mathcal{O}[\![S(x_1)]\!] \neq \mathcal{O}[\![S(x_2)]\!]])]$$

where $Cont^{\mathcal{D}} = \{(\alpha, [\![k]\!]) \mid (\alpha, k) \in CRes\}$ is the class of denotable continuations for $[\![\cdot]\!]$. Suppose $x_1, x_2 \in X$, $(\alpha, k) \in CRes$ are such that $[\![x_1]\!](\alpha, [\![k]\!]) \neq [\![x_2]\!](\alpha, [\![k]\!])$. By Lemma 7.9 and Remark 7.10(b) there exists an $\mathcal{L}$ syntactic context S such that $\mathcal{D}[\![S(x_1)]\!] = [\![x_1]\!](\alpha, [\![k]\!]) \neq [\![x_2]\!](\alpha, [\![k]\!]) = \mathcal{D}[\![S(x_2)]\!]$. From [27] we know that $\mathcal{D}[\![x]\!] = \mathcal{O}[\![x]\!], \forall x \in X$. Hence, $\mathcal{O}[\![S(x_1)]\!] = \mathcal{D}[\![S(x_1)]\!] \neq \mathcal{D}[\![S(x_2)]\!] = \mathcal{O}[\![S(x_2)]\!]$, i.e. there exists an $\mathcal{L}$ syntactic context S such that $\mathcal{O}[\![S(x_1)]\!] \neq \mathcal{O}[\![S(x_2)]\!]$. The conclusion is that $[\![\cdot]\!]$ is (weakly complete and thus) weakly abstract with respect to $\mathcal{O}[\![\cdot]\!]$. □

8 Concluding remarks and future research

We investigate semantic properties in denotational models designed with metric spaces and continuation semantics for concurrency (CSC) [27]. For illustration purposes we considered a simple concurrent language embodying the paradigm of asynchronous communication introduced in [8]. For the language under consideration we established various semantic properties in a *class of denotable continuations* and a corresponding *domain of denotable continuations* (obtained by metric completion from the class of denotable continuations).

We introduced an abstraction criterion - that we call *weak abstractness* - which relaxes the completion condition of the classic full abstractness criterion introduced by Robin Milner [16] [17]. We proved that the denotational model presented in this paper is weakly abstract with respect to a corresponding operational model (initially introduced in [27]). The weak abstractness condition was established for CSC. We do not know whether a fully abstract domain of CSC exists. The full abstracness problem seems to be particularly difficult in continuation semantics. We are not aware of any full abstractness result for a denotational model designed with continuations for concurrent languages, although various papers employ continuations in the denotational description of concurrent languages [3], [2], [24], [10].

We aim to obtain similar weak abstractness results for other concurrent languages designed semantically with CSC, including the Basic Andorra Model [30], and and CSP-like synchronous communication on multiple channels [28] in the style introduced in the Join calculus [14]. It may also be useful to investigate similar (weak) abstractness criteria for other domains of continuations, especially where full abstractness results are difficult (or impossible) to achieve.

References

[1] P. America and J.J.M.M. Rutten, Solving reflexive domain equations in a category of complete metric spaces, *J. of Comp. Syst. Sci* 39(3), 343–375, 1989.

[2] J.W. De Bakker and E.P. De Vink, Rendez-vous with metric semantics, *New Generation Computing* 12, 53–90, 1993.

[3] J.W. De Bakker and E.P. De Vink: *Control Flow Semantics*, MIT Press, 1996.

[4] J.C.M. Baeten and W.P. Weijland, *Process algebra,* Cambridge Univ. Press, 1990.

[5] J.W. De Bakker and J.H.A. Warmerdam, Metric pomset semantics for a concurrent language with recursion, *LNCS* 469, 21–49, Springer, 1990.

[6] J.A. Bergstra and J.W. Klop, Algebra of communicating processes with abstraction, *Theoretical Computer Science* 37(1), 77–121, 1985.

[7] D.G. Bobrow and B. Wegbreit, A Model and Stack Implementation of Multiple Environments, *Comm. ACM* 16(10), 591–603, 1973.

[8] De Boer, F.S., Kok, J.N., Palamidessi, C., Rutten, J.J.M.M.: A paradigm for asynchronous communication and its application to concurrent constraint programming. In Apt, K.R., De Bakker, J.W. and Rutten, J.J.M.M, eds., "Logic Programming Languages: Constraints, Functions and Objects", MIT Press, 82-114 (1993)

[9] S. Brookes, Traces: a unifying semantic framework for parallel programming languages, *MFPS* 18, New Orleans, 2002.

[10] A. De Bruin and W. Bohm, The denotational semantics of dynamic networks of processes, *ACM Transactions on Programming Languages and Systems* 7(4), 656–679, 1985.

[11] G. Ciobanu and E.N. Todoran, Continuation semantics for asynchronous concurrency, Technical Report FML-10-02, Romanian Academy, 2010. Available at `http://iit.iit.tuiasi.ro/TR/reports/fml1002.pdf`

[12] G. Ciobanu and E.N. Todoran, Continuation semantics for asynchronous concurrency, *Fundamenta Informaticae*, to appear.

[13] O. Danvy, On evaluation contexts, continuations and the rest of the computation, *4th ACM SIGPLAN Continuations Workshop*, 13–23 2004.

[14] C. Fournet and G. Gonthier, The Join calculus: a language for distributed mobile programming, *LNCS* 2395, 268–332, 2002.

[15] Honda, K., Tokoro, M.: An object calculus for asynchronous communication, *LNCS* vol.512, 133–147, Springer (1991)

[16] R. Milner. Processes: a mathematical model of computing agents. In *Proc. of Logic Colloquium*, pages 157-173, North Holland, Amsterdam, 1975.

[17] R. Milner. Fully abstract models of typed λ-calculi. *Theoretical Computer Science*, 4:1–22, 1977.

[18] R. Milner, A calculus of communicating systems, *LNCS* 92, Springer, 1980.

[19] R. Milner, *Communication and concurrency*, Prentice Hall, 1989.

[20] R. Milner. *Communicating and mobile systems: the π calculus*. Cambridge University Press, 1999.

[21] C. Palamidessi, Comparing the expressive power of the synchronous and the asynchronous π calculus, *Math. Structures in Computer Science*, 13(5): 685–719, 2003.

[22] G. Plotkin, A powerdomain construction, *SIAM Journal of Computing* 5(3), 452–487, 1976.

[23] G. Plotkin. A Structural Approach to Operational Semantics, *Journal of Logic and Algebraic Programming*, 60-61: 17–139, 2004.

[24] J.J.M.M. Rutten, Semantic correctness for a parallel object oriented language, *SIAM Journal of Computing* 19(2), 341–383, 1990.

[25] V. Saraswat, *Concurrent constraint programming*, MIT Press, 1993.

[26] C. Stratchey and C. Wadsworth, Continuations: a mathematical semantics for handling full jumps, *Higher-Order and Symbolic Comput.* 13, 135–152, 2000.

[27] E.N. Todoran, Metric semantics for synchronous and asynchronous communication: a continuation-based approach, *ENTCS* 28, 119–146, Elsevier, 2000.

[28] E.N. Todoran, Comparative semantics for modern communication abstractions, *Proc. IEEE ICCP'08*, 153–160, 2008.

[29] Todoran, E.N.: Metric semantics for modern second order communication abstractions. *Proc. IEEE ICCP'09*, 215–219, 2009.

[30] E.N. Todoran and N. Papaspyrou, Continuations for parallel logic programming, *Proc. ACM PPDP'00*, 257–267, ACM Press, 2000.

[31] *Synchronous and Asynchronous Interaction in Distributed Systems (SAS)*, Project funded by DFG (German Research Foundation), 2010, `http://concurrency-theory.service.tu-berlin.de/joomla/projects/projects`.

Some technical Lemmas

The proofs of Lemmas .1 and .2 are left to the reader. They can be approached by the identification of appropriate invariant properties in combination with uses of the '$\varepsilon \leq \frac{1}{2} \cdot \varepsilon \Rightarrow \varepsilon = 0$' argument.

Lemma .1 *For all* $x_0, x_1, x_2 \in X, \sigma \in \Sigma, \alpha_0, \alpha \in Id, k \in KRes$ *such that* $(\alpha_0, k) \in CRes, \nu(\alpha_0, k), \alpha \notin id(k), \alpha \cdot 1 \notin id(k)$ *and* $(\neg(\alpha_0 \leq \alpha \cdot 1))$ *we have:*

$$[\![x_0]\!](\alpha_0, [[\![k]\!] \mid \alpha \mapsto [\![x_1; x_2]\!]])(\sigma) =$$

$$[\![x_0]\!](\alpha_0, [[\![k]\!] \mid \alpha \cdot 1 \mapsto [\![x_1]\!] \mid \alpha \mapsto [\![x_2]\!]])(\sigma)$$

Lemma .2 *For all* $x_0, x_1, x_2 \in X, \sigma \in \Sigma, \alpha_0, \alpha \in Id, k \in KRes$ *such that* $(\alpha_0, k), (\alpha, k) \in CRes, \nu(\alpha_0, k), \nu(\alpha, k)$, $(\alpha_0 \neq \alpha), \neg(\alpha_0 \leq \alpha)$ *and* $\neg(\alpha \leq \alpha_0)$ *we have:*

$$[\![x_0]\!](\alpha_0, [[\![k]\!] \mid \alpha \mapsto [\![x_1 \parallel x_2]\!]])(\sigma) =$$

$$[\![x_0]\!](\alpha_0, [[\![k]\!] \mid \alpha \cdot 1 \mapsto [\![x_1]\!] \mid \alpha \cdot 2 \mapsto [\![x_2]\!]])(\sigma)$$

In Definition .3 we introduce two predicates *seq* and *par* that express conditions for sequential and parallel computations, respectively. Some properties of continuations and resumptions expressed based on these predicates are stated in Lemma .4, whose proof is left to the reader.

Definition .3 *We define* $seq, par : (A \times A \times \mathcal{P}_{fin}(A)) \to Bool$ *as follows*

- $seq(\alpha', \alpha'', \pi) = (\alpha', \alpha'' \in \pi)$ and $(\alpha' < \alpha'')$ and
$$(\forall \overline{\alpha} \in \pi : ((\alpha' < \overline{\alpha}) \Rightarrow (\alpha'' \leq \overline{\alpha})))$$
- $par(\alpha', \alpha'', \pi) = (\alpha', \alpha'' \in max(\pi))$ and $(\alpha' \neq \alpha'')$ and
$$(\forall \overline{\alpha} \in \pi : ((glb(\alpha', \alpha'') < \overline{\alpha}) \Rightarrow$$
$$((\neg(\overline{\alpha} < \alpha') \text{ and } (\neg(\overline{\alpha} < \alpha'')))))$$

Here $glb(\alpha', \alpha'')$ *is the greatest lower bound of* α', α'' *with respect to the partial order* $\leq$ *defined on A (see Definition 4.1); also, we use the notation* $\alpha' < \alpha''$ *to express that* $\alpha' \leq \alpha''$ *and* $\alpha \neq \alpha''$.

Lemma .4

(a) If $\pi \in \mathcal{P}_{fin}(A)$, $|\pi| > 1$, *then* $\exists \alpha', \alpha'' \in \pi$ *such that:* $seq(\alpha', \alpha'', \pi)$ or $par(\alpha', \alpha'', \pi)$.

(b) If $(\alpha, k) \in CRes$, $\nu(\alpha, k)$, $|id(k)| \geq 1$ *and* $seq(\alpha', \alpha'', \alpha \cup id(k))$ *then:*

(i) either $\alpha'' = \alpha$ *and* $(\alpha, k) \cong (\alpha' \cdot 1, k)$

(ii) *or* $\alpha'' \neq \alpha$ *and* $\alpha'' \geq \alpha' \cdot 1$ *and* $(\alpha, k) \cong (\alpha, [k \setminus \{\alpha''\} \mid \alpha' \cdot 1 \mapsto k(\alpha'')])$,

(iii) *or* $\alpha'' \neq \alpha$ *and* $\alpha'' \geq \alpha' \cdot 2$ *and* $(\alpha, k) \cong (\alpha, [k \setminus \{\alpha''\} \mid \alpha' \cdot 2 \mapsto k(\alpha'')])$.[10]

(c) *If* $(\alpha, k) \in CRes$, $\nu(\alpha, k)$, $|id(k)| \geq 1$ *and* $par(\alpha', \alpha'', \{\alpha\} \cup id(k))$ *then:*

(i) *either* $\alpha' = \alpha$ *and* $(\alpha, k) \cong (\alpha' \cdot 1, [k \setminus \{\alpha''\} \mid \alpha' \cdot 2 \mapsto k(\alpha'')])$

(ii) *or* $\alpha'' = \alpha$ *and* $(\alpha, k) \cong (\alpha'' \cdot 1, [k \setminus \{\alpha'\} \mid \alpha'' \cdot 2 \mapsto k(\alpha')])$

(iii) *or* $\alpha' \neq \alpha$ *and* $\alpha'' \neq \alpha$ *and*

$$(\alpha, k) \cong (\alpha, [k \setminus \{\alpha', \alpha''\} \mid \alpha' \cdot 1 \mapsto k(\alpha') \mid \alpha' \cdot 2 \mapsto k(\alpha'')])$$

[10] $\alpha'' \neq \alpha$ implies $\alpha' \neq \alpha$, because $\nu(\alpha, k)$ and $\alpha' < \alpha''$.